The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Biomechanical Joint Motion Synthesis" submitted by Christopher D. Thorne in partial fulfilment of the requirements for the degree of Master of Science.

Supervisor, Brian Wyvill
Department of Computer Science

Przemyslaw Prusinkiewicz
Department of Computer Science

Bruce MacDonald
Department of Computer Science

Ton van den Bogert
Department of Biomechanics

April 12, 1995
Date
Abstract

Joint motion studies are important from a medical viewpoint since successful operation outcome has been correlated with normal joint motion. Often biomechanical models are devised for these studies in clinical or non-clinical environments. Recently, a model has been proposed in which joint motion is predicted from one magnetic resonance scan. The new model of joint motion will be evaluated in this research.

Volume data is first acquired from a medical scanner, and the joint is segmented into cartilage structures. The motion of the joint is then simulated by using an optimization technique to compute a path of maximum contact area. The simulated joint motion matches empirical observations, but further work is required to increase the agreement with biomechanics literature.
Acknowledgments

Ever since I was a young lad Computer Science has been an interest of mine. I cannot succinctly explain my reasons for pursuing this degree; but perhaps knowledge and freedom of research best explain my motives. A wise professor once told me graduate school was simple — two years of relaxation, I believe were the words. Another professor said that being a graduate student is the best part of your life. I am not at an age to confirm this statement, but I will admit that the time spent at the University of Calgary has been great. There has been so many things to do and so little time. I recall seeing many bands, playing games of squash, ice hockey, skating during lunch time with colleagues, entertaining myself with a soap opera of which I was part of in ballroom dance, early morning weight lifting with Earle, going for beverages at the Grad Lounge, volunteering at CJSW, and countless days of downhill skiing.

I would like to thank my office colleagues who have also endured graduate student life: Sean Brandenburg, Dale Brisinda, Andy Brownbill, Kevin Chmilar, Leon Fedenczuk, Carl Gutwin, Andy Guy, Mark Hammel, Earle Lowe, Radomír Mêch, Ted O'Grady, Mark Roseman, Doug Schaffer, and Hongwen Zhang. Many late nights have been spent bonding with these people to meet deadlines. I would also like to give special thanks to the patients of the biomechanical study, Bruce MacDonald for his outstanding assistance and supervision of my thesis corrections, Steve Franks for reviewing the first draft of my thesis, Danny Dimian for providing his personal copy of the FrameMaker manual over Christmas holidays, and Gordon Hamilton for his expertise in mathematics and assistance in this research. Most importantly, I would like to thank my parents and Challen Pride for their unsurpassed support.
# Table of Contents

Approval Page ........................................ ii

Abstract ............................................. iii

Acknowledgments ....................................... iv

Table of Contents ...................................... v

List of Tables .......................................... viii

List of Figures ......................................... ix

Chapter 1 Introduction .................................. 1
  1.1 Statement of Thesis .................................. 1
  1.2 Contributions ....................................... 2
  1.3 Thesis Organization ................................ 3

Chapter 2 Literature Survey ............................ 4
  2.1 Computer Graphics Representations .................. 5
    2.1.1 Surface Representations ......................... 6
    2.1.2 Solid Representations ............................. 7
    2.1.3 A comparison of surface and solid representations 7
  2.2 Volume Modeling .................................... 8
  2.3 Volume Rendering .................................. 10
    2.3.1 Indirect Rendering ............................... 10
    2.3.2 Direct Rendering ................................ 11
    2.3.3 A comparison of surface and volume rendering techniques 12
  2.4 Animation ......................................... 13
    2.4.1 Key-frame Systems .............................. 13
## List of Tables

1. Results after resampling .................................................. 23
2. Numeric results from volume intersection .......................... 53
3. Frequency of ambiguous polygonalization ........................ 60
4. Results for *ContactArea* example one ............................. 61
5. Results from *ContactArea* example two ....................... 62
6. Exhaustive sampling interval for synthetic example one ....... 71
7. Exhaustive sampling interval for synthetic example two ...... 74
8. Empirical observations of subtalar joint movement through a horizontal and roll step 76
List of Figures

1 Anatomy of the subtalar joint ........................................... 18
2 The three orthogonal planes of volume data ................................................. 20
3 Reformatting medical data of the subtalar joint ..................................................... 22
4 Segmentation of a two dimensional volume into two structures .............................................. 25
5 Segmentation using region growing ................................................................. 28
6 Segmentation using region growing with distance .................................................... 29
7 Segmentation using region growing with a threshold ................................................. 30
8 Creating a colour map with spline curves ............................................................. 31
9 A comparison of constant and smooth shading ....................................................... 31
10 Three examples of adjusting the transparency value to blend a voxel’s value and classification colour ............................................................. 32
11 Three examples of increasing the selected structure’s brightness ........................................... 33
12 Three examples of magnifying an image ............................................................... 34
13 Segmentation through reformatted projection .......................................................... 34
14 A map of the pre-processing pipeline ................................................................. 35
15 The segmentation map ...................................................................................... 35
16 Energy function for computing path of motion ......................................................... 39
17 The subtalar joint with labeled constraints .............................................................. 40
18 Horizontally stepping the talus ........................................................................ 41
19 Subtalar motion shown as primarily left to right translation ........................................... 41
20 The algorithm for path of motion simulation .......................................................... 43
21 The algorithm for Fletcher–Reeves optimization ....................................................... 45
22 An example of vertical displacement to achieve intersection ........................................... 47
23 Algorithm to resolve volume constraints ............................................................... 47
24 Graph of the percentage intersection function ......................................................... 52
25 Volume intersection example one ..................................................................... 54
26 Volume intersection example two ..................................................................... 54
27 Estimating contact with arc length .................................................................... 55
28 Using set theory to compute arc length. ........................................... 56
29 The twenty six, eighteen, and six neighborhood perimeters. ............... 57
30 Two possible configurations for lattice polygonalization. .................... 59
31 Four positions of interest for ContactArea example one ..................... 61
32 Two positions for ContactArea example two .................................... 62
33 The FletcherReeves dialog ............................................................... 64
34 The FletcherReeves map ................................................................. 64
35 The ContactVolume dialog ............................................................. 65
36 The ContactVolume map ................................................................. 66
37 The ContactArea dialog ................................................................. 67
38 An Explorer map to compute contact area ........................................ 67
39 Frames zero, eight, and fifteen of synthetic example one .................... 69
40 Graph of optimized positions for synthetic example one ..................... 70
41 Volume representation of the sphere .................................................. 70
42 Graph of linear search positions for synthetic example one ................. 71
43 Frames zero, seven, and fifteen of synthetic example two ................... 72
44 Graph of optimized positions for synthetic example two ..................... 72
45 Coordinate system of the ellipsoid .................................................... 73
46 Polygonization of the fourth frame .................................................... 73
47 Graph of Hamilton's criteria for synthetic example two ....................... 74
48 Variations in position of axis as projected upon sagittal plane. ............. 77
49 Variations in position of axis as projected upon transverse plane .......... 77
50 Coordinate system of the talus cartilage .......................................... 78
51 Graph of positions obtained by stepping horizontally and optimizing Hamilton's criteria ................................................................. 78
52 Graph of positions obtained by stepping horizontally and optimizing contact area ................................................................. 79
53 Graph of Hamilton's criteria obtained by stepping horizontally ............. 80
54 Graph of contact area obtained by stepping horizontally ..................... 80
55 Graph of positions obtained by stepping roll and optimizing Hamilton's criteria ................................................................. 81
56 The talus cartilage centre of mass ..................................................... 82
Chapter 1

Introduction

*Range of motion* is the set of joint positions that can be achieved. It has important applications in the clinical situation because the success of operations on fractured joints has been correlated to normal range of motion [Paley and Hall 1989]. In the past, measuring a joint’s range of motion required actual motion. Physicians manually moved the joint and compared the range of motion with the *contralateral* (opposite) limb. The primary problem with this measurement is accuracy. Range of motion has also been measured in the non-clinical environment through the use of *in-vitro* (outside the living body) and *in-vivo* (inside the living body) studies. These studies require the attachment or implantation of markers or are ethically problematic. All aforementioned studies require joint movement.

In the clinical situation following reduction, the bringing of displaced or broken parts back into their normal positions, a patient will be unable and unwilling to move their joint due to swelling and pain. Furthermore, treatment planning and reduction assessment will be difficult to evaluate because the joint’s range of motion cannot be determined. *Biomechanical Joint Motion Synthesis* addresses this problem by simulating joint motion from volume scan data without joint movement.

1.1 Statement of Thesis

Joint motion is a problem commonly tackled in the field of biomechanics, and computer science is often applied to assist research because some calculations require numerical techniques. In this case, a biomechanics researcher has proposed that any joint position can be assessed numerically for membership in the range of motion, by optimizing the
surface area of cartilage contact with a constant volume of intersection, under a model in which cartilage shape is fixed. The goal of this research is to validate the empirical validity of this biomechanics supposition. In the process, software tools will be developed to evaluate joint positions and compute the path of motion of the subtalar (ankle) joint. This thesis will be concerned with paths in the range of motion. In particular I will prescribe one degree of freedom of the motion and find a path by optimizing the contact area over other degrees of freedom. A path of motion in only a subset of the range, and may not give joint limits along other paths, however in future work, range of motion could be assessed from paths or a single carefully chosen path of motion. The predicted path will be evaluated by comparing the simulated behavior with empirical observations and biomechanics literature that describes the motion of the subtalar joint.

1.2 Contributions

The main contribution of this work is in verifying a biomechanics model for joint motion. Results obtained roughly match scientific observations, but are not in agreement with biomechanics literature. Specific areas of future work are outlined.

Key--frames, kinematics, dynamics, or inverse dynamics have previously been used in computer graphics. These techniques are well suited if the object’s motion is known or can be mathematically described; however, with some biological joints this is not the case. A new technique to simulate motion of volume objects is presented.

This work also contributes to the field of biomechanics by using a volume instead of a traditional surface representation. Volume modeling is justified because some objects are difficult to describe with surfaces. Furthermore, intersection calculation --- which is an important part of motion synthesis --- is easily solved.
1.3 Thesis Organization

Chapter 2 Literature Survey, is a short overview of computer graphics modeling, volume rendering, and animation techniques. Specifically, the modeling and rendering methods discussed have expressly been used in this work. Several algorithms for motion synthesis are also presented, and the purpose of this discussion is to give background for understanding why Biomechanical Joint Motion Synthesis is different from previous approaches.

Chapter 3 Modeling, presents a collaboration of miscellaneous modules to preprocess, and segment volume data. Most of this section is devoted to a description of the segmentation tool. It has been developed to facilitate modeling biological structures from imaged data. This tool makes use of several computer graphics techniques including: compositing, reformatting, and colour space transforms.

Chapter 4 Motion Simulation, comprises the bulk and primary contributions of this thesis. The simulation is controlled by an optimization process which associates a numeric measure with each joint position. The measure — which is based on recent biomechanics research — is comprised of estimating volume of intersection and contact area. A new technique to compute these properties is presented.

Chapter 5 Results, the results from executing the simulation on synthetic and medical data are shown. The simulated path of motion matches empirical observations; but not biomechanics literature.
Chapter 2

Literature Survey

The last twenty years has seen the advent of medical imaging systems that can non-invasively generate accurate volume information of living patients. These technologies include computed tomography (CT), magnetic resonance imaging (MRI), ultrasound, and nuclear medicine modalities such as positron emission tomography (PET) and single photon emission computed tomography (SPECT). The rapid expansion in both the development of three-dimensional medical technologies and their widespread use has generated new frontiers for research with computers in medicine. Biomechanical Joint Motion Synthesis is one of the new directions of research that uses these imaging technologies for non-invasive studies of joint motion.

Traditionally, medical imaging has been used to improve and expand diagnostic capability with respect to the identification, classification, and localization of pathological processes. These imaging technologies have also been used to enhance therapeutic interventions such as stereotaxis biopsy, surgical planning and radiation dosimetry. Two and three-dimensional imaging have been used for a wide range of orthopedic applications. Some of the more common applications include pelvic and acetabular (hip joint) fractures [Scott et al. 1987; Fishman et al. 1986; Magid et al. 1985; Ney et al. 1989; Pate et al. 1986; Burk et al. 1985], trauma to the lower extremity (knee and ankle) [Magid et al. 1990; Vannier et al. 1985], spine [Sunburg et al. 1986; Wojcik et al. 1987; Kilocyne and Mack 1987], and upper extremity (the shoulder complex) [Kuhlman et al. 1988]. Three dimensional imaging has also been used in several biomedical applications [Farrell and Zappulla 1989] including visualization of bacteriophage, chromosome structures, cells, embryonic heart, and primate neo-cortex. Tumor growth and primate biology from fossil skulls have also
been studied through medical imaging.

This previous work has demonstrated that information gained from volumetric rendering of imaged data can make a significant impact on patient management. The rendered images can represent articulated bones from all perspectives, but the joint itself remains depicted in one position and the joint's functional range of motion is not shown. Range of motion could be studied by gathering multiple images of the joint in different positions, but this is not feasible due to excessive radiation (for some scanning technologies) and limited patient tolerance. In [Gerber et al. 1991] a technique termed simulated dynamic CT animates the range of motion of the femur in the acetabulum. Their work incorporates static CT data with gait and movement data to produce a sequence of dynamic images simulating range of motion.

In this work, path of motion is computed by numerically accessing joint positions. This method of simulating motion differs from the work of [Gerber et al. 1991]. However, they both share common problems associated with modeling the joint by a computer graphics representation, constructing the model from volume data (for example, magnetic resonance imaging), and rendering the model. Simulated dynamic CT and other medical applications have used a solid representation, volume modeling, and volume rendering to solve each of these problems respectively. In this chapter these topics are presented.

The third component of this research is simulating joint motion to produce an animation. Computer graphics presents several methods to predict or synthesize motion and they are presented in this chapter as well. Of these animation techniques, motion synthesis through optimization is the most relevant towards the requirements of this work.

### 2.1 Computer Graphics Representations

A model is a representation of some (not necessarily all) features of a concrete or abstract entity. The purpose of a model is to allow people to visualize and understand the structure
or behavior of the entity and to provide a convenient vehicle for "experimentation" with and prediction of the effects of inputs or changes to the model. Often models simplify the actual structure or behavior of the modeled entity to make it easier to visualize, or for those models represented by systems of equations, to make it computationally tractable. Graphics can be used to create and edit the model, to obtain values for its parameters, and to visualize its behavior and structure. In the context of this work, the term model refers a computer graphics representation of the joint and modeling refers to the steps required to construct the model.

In this section, two different computer graphics representations are discussed. The representation is important because it affects: the data structure used, the type of processing algorithms (for example, computing intersection between two models, or model rendering), the cost of processing the model (for example, memory use or computation time), and the model's ease of modification or construction.

Surface representations will first be discussed in Section 2.1.1, and solid representations are then presented in Section 2.1.2. The advantages and disadvantages of these representations are then discussed in Section 2.1.3.

2.1.1 Surface Representations

Today, surfaces are the mainstay of modeling objects in computer graphics. Several algorithms deal with surfaces, and many workstations are specially designed to process and render surfaces and their polygonal approximations.

Surface modeling is quite broad, but the three most common representations for three dimensional surfaces are: polygon mesh, parametric bivariate (two-variable) polynomial surface patches, and quadric surfaces. Open boxes, cabinets, and building exteriors can be easily and naturally represented by polygon meshes. Curved objects can also be modeled with polygon meshes; however, less bivariate surface patches are required to approximate
the curved object to a given accuracy. Lastly, quadric or implicit surfaces are a convenient representation for the familiar sphere, ellipsoid, and cylinder. While this class of shapes is somewhat restricted, it can be used to advantage in modeling many useful objects by combining the primitives to form more complex objects. Implicit equations are not only restricted to modeling surfaces, they also be used to model solids. For example, the implicit equations to model the surface and solid of a unit sphere respectively are:

\[ x^2 + y^2 = 1 \quad \text{and} \quad x^2 + y^2 \leq 1 \]

### 2.1.2 Solid Representations

A representation’s ability to encode things that look like solids does not by itself mean that the representation is adequate for representing solids. In two dimensions, lines and curves describe the boundary of a closed area and not the contents of the area. Similarly, in three dimensions, planes and surfaces describe the boundary of a closed volume and not the contents of the volume. In many applications, it is important to distinguish between the inside, outside, and surface of a three dimensional object and to be able to compute properties (for example, volume, and centre of mass) of the object that depend on this distinction. A variety of specialized ways of representing these objects have been developed: primitive instancing, sweep methods, boundary models, spatial enumeration, cell decomposition, implicit solids, and constructive solid geometry.

### 2.1.3 A comparison of surface and solid representations

Solid representations have advantages over surface representations in that intersection — an important component of motion simulation — can easily be determined. In general, surface to surface intersection remains a challenging computational problem, and much research is still required to achieve a good balance between the conflicting goals of reliability, accuracy, and efficiency in algorithms attempting to solve it [Patrikalakis 1993]. On the other hand, intersection is straightforward with many solid representations. As are other motion simulation measurements (for example, contact area and volume of intersec-
tion) because they can be discretely approximated. From this discussion, spatial enumeration (a solid representation) has been chosen based on ease of model construction and implementation of these calculations.

In this representation, the solid is decomposed into identical cells arranged in a fixed regular grid. These cells are called voxels in analogy to pixels. When representing an object using spatial occupancy enumeration, we control only the presence or absence of a single cell at each position in a grid. To represent an object we need only decide which cells are occupied and which are not. The object can thus be encoded by a unique and unambiguous list of occupied cells. It is easy to find out whether a cell is inside or outside the solid and determining whether two objects are adjacent is simple as well.

2.2 Volume Modeling

Volume modeling is a process of constructing a surface or solid model from volume data. The data can be generated by a computer simulation (for example, in computational fluid dynamics), by voxelizing a geometric model (for example, a sphere), or by sampling a real object with a scanner (for example, a real human organ sampled by a magnetic resonance scanner). Volume modeling is then the synthesis, analysis, and manipulation of shapes contained within a volume of data. These shapes can be physical, representing anatomical objects (for example, hips, ankles, brains), or abstract showing mathematical surfaces such as electron density or constant velocity and pressure.

Medical applications use numeric or image data from experiments in such fields as computed tomography, magnetic resonance, and positron emission tomography. Each point in the three dimensional space represents an intensity averaged over a small volume and is termed a voxel. In order to effectively use this information hidden within the three dimensional data set, the data must first be processed so that sub-regions of interest within the data are identified.
Hanrahan [Hanrahan 1993] has used the term “model–based segmentation” to describe the step of extracting meaningful objects from acquired physical data. The term segmentation is usually applied to medical data where it signifies the process of identifying which pixels or voxels of an image belong to a particular object (for example, air, bone, fat, or tissue). This same segmentation and feature–extraction process is also of potential interest for other kinds of data such as identifying the geometry of certain subterranean objects in geophysical data. A wide variety of volume modeling methods have been developed to solve this problem and they can be grouped into three principal approaches based on the primitive used for grouping voxels together.

In the first approach, contours are used to group voxels. Contours are formed either by manually tracing structures of interest or by automatic edge tracking based on thresholds. In the next approach, the voxels of interest are bounded by a surface. One approach is based on tiling between contours to form a three dimensional surface [Fuchs et al. 1977]. A surface can also be defined by tracking the boundary of a binary region defined by value thresholds [Wyvill et al. 1986]. Lastly, the voxels or volume primitives themselves can be used to represent the model by either tree encoding (for example, octrees) [Samet and Webber 1988], surface selection (with the volume’s perimeter voxels), run–length encoding [Reynolds et al 1987], or straight data array [Herman et al. 1988]. In [Shirley and Neeman 1989] two additional methods based on the binary array are used. The first is a rectangular lattice where the spacings may be uneven. This is useful in applications where the cell density increases in regions of interest. In the second method, the volume is a connected lattice of irregularly shaped cells (a curvilinear lattice).

The tree encoding can be used to reduce data storage; however, several steps are required to obtain a tree description before display. Contours and surfaces also result in a large reduction in data storage and certain display functions can be performed rapidly. On the other hand significant preprocessing is needed to form the surfaces or contours, and an accurate surface representation of complex or convoluted objects may not be feasible. In comparison, the volume approach — which has been chosen for this work — does not
require significant pre-processing before the data set is accessed. The user can view any part of the data without the pre-processing needed for contours or surfaces. The volume primitive methods are not limited by structure complexity and do not depend on the assumption of a structure. Data storage and retrieval are the important limiting factors of this approach.

2.3 Volume Rendering

The process of creating images from models is often called rendering.

Volume rendering today has progressed from its initial applications in medical imaging to reconstructing scattered data and representing geometrical objects without mathematically describing the surfaces (for example, clouds, water molecules, and other phenomena from both empirical and simulated scientific data). Volume rendering can be used to show the characteristics of the interior of a solid region in a two dimensional image. In a typical medical example, after imaging a human or animal the display of this information can indicate the various types of tissue (as indicated by density changes), and boundaries between structures.

Several approaches to volume rendering are discussed and they can be divided into two categories: those that compute level surfaces — indirect rendering (Section 2.3.1), and those that display integrals of density along rays — direct rendering (Section 2.3.2). Indirect rendering is discussed because these techniques have been used to measure volume properties (for example, surface area), and produce the volume images shown in this thesis. The direct rendering techniques are presented for completeness of a volume rendering survey.

2.3.1 Indirect Rendering

Surface rendering is an indirect technique for visualizing volume primitives by first con-
verting them into an intermediate surface representation and then using conventional computer graphics techniques to render them to the screen.

Volume data is first converted into geometric primitives in a process called iso-surfacing, iso-contouring, surface extraction, or border following [Herman and Liu 1979; Cook et al. 1983; Wyvill et al. 1986; Cline et al. 1988]. In the iso-surfacing algorithm scalar values are assumed to be given at each point of a lattice in a three dimensional data set. A particular level surface is approximated by determining all intersections of the level surface with edges of a lattice. Pairs of adjacent lattice points whose field values surround the desired value are examined, and the location of an intersection of the level surface with the edge is then estimated by linear interpolation. The collection of all the surface pieces just defined constitutes a surface, and the resulting level surface can be rendered with conventional techniques. This strategy can be of use in medical imaging to show the shape of the boundary between different types of tissue.

2.3.2 Direct Rendering

In comparison to indirect or surface rendering, volume rendering is a direct display of volume primitives without any intermediate conversion of the volume data to a surface representation.

Upson and Keeler [Upson and Keeler 1988] present two methods for the display of scalar fields (a volume data set of one variable). In both methods the voxel’s scalar value is associated with four different functions for: red, green, blue, and opacity. The voxel’s colour is obtained by evaluating these functions with the voxel’s scalar value. The first rendering method is based on a ray-tracing mechanism for volume data [Kajiya and Von Herzen 1984]. For each ray from the eyepoint through an image pixel, the red, green, blue, and opacity values of the voxels are accumulated as the ray passes through the volume data. This accumulation stops when the opacity reaches a maximum or the ray exits the volume. In their second approach, the same basic notion of integration along rays is used, but the
voxels are projected in a front-to-back order.

Sabella [Sabella 1988] also makes use of a ray-casting algorithm. Each point in the volume is characterized as a varying density emitter (a system of particle light sources). The light reaching the eye along any ray is computed by summing up the emission from all the emitters along the ray, mapping the results to various colour scales, and then attenuating the light by the Lambert lighting model to simulate scattering.

Drebin, Carpenter, and Hanrahan [Drebin et al. 1988] present a technique for rendering images of volumes containing mixtures of materials (for example, bone, fat, and soft tissue). First, each voxel is converted by a material classification process into a *material percentage volume*, which is used in calculating colour and partial opacity. Each voxel is then projected onto an image plane and blended together with the projection formed by previous slices. Through this process, the final image is produced by a back-to-front composite of transformed slices.

### 2.3.3 A comparison of surface and volume rendering techniques

Surface rendering results in a significant reduction in data storage and can exploit existing software and geometry engines for fast manipulation and display. However, substantial pre-processing is required to extract the surfaces from the volumetric data. Surface rendering assumes that the data consists of tangible surfaces that can be extracted and visualized, but for some complex or gel-like objects, it is not possible to form a surface representation (for example, seismic data). Furthermore, surfaces remove the information inside the three dimensional objects, and visceral exploration is no longer available because the only embodiment of the object is the surface itself.

Volume rendering maintains the entire data set, and thus any part, including parts interior to the objects, can be viewed. Volume rendering techniques are structure independent, and thus they are more effective for complex scenes. Volume rendering does not require sub-
stantial pre-processing, although relevant components need to be identified during a pre-processing step. Volume storage and processing time are the current limitations of the volume rendering approach.

2.4 Animation

Animation is used widely in the entertainment industry, and is also being applied in education, in industrial applications such as control systems and heads-up displays and flight simulators for aircraft, and in scientific research. The scientific applications of computer graphics, and especially of animation, have come to be grouped under the heading scientific visualization. Visualization is more than the mere application of graphics to science and engineering; it can involve other disciplines, such as signal processing, computational geometry, and database theory. Often, the animations in scientific visualization are generated from simulations of scientific phenomena. The results of the simulations may be large data sets representing two or three dimensional data (for example, fluid-flow simulations); this data is converted into a series of images that constitute the animation. At the other extreme, the simulation may generate positions and locations of physical objects, which must then be rendered in some form to generate the animation.

The techniques presented in this section include: key-frame systems, inverse kinematics, dynamics, and optimal control. These methods are not appropriate for studying range of motion because the joint's motion must be mathematically described or known before animation. This work determines range of motion where no description existed before.

2.4.1 Key-frame Systems

Key-frame animation is a traditional technique where an animation sequence is derived by in-betweening two drawings depicting the start and end frames. These frames must have a reasonable level of coherence, otherwise the in-between frames will not smoothly reflect a transition from the start to the end. In computer animation, variables can be inter-
polated, or images can be warped [Beier and Neely 1992]. Key–frame animation is therefore widely used in computer animation, and most professional systems provide facilities whereby an animator can specify those parameters to be interpolated.

2.4.2 Inverse Kinematics

Kinematics refers to the positions and velocities of points or objects. A kinematic description of a scene, for example, might be, “The cube is at the origin at time \( t = 0 \), and it moves with a constant acceleration in the direction \( (1, 1, 5) \).” This is called forward kinematics; however, this is not much use to an animator because there is no explicit control over the final position. With inverse kinematics the above scene description becomes, “What must the velocity of the cube be for it to reach position \( (12, 12, 42) \) in 5 seconds?” For simple systems, these sorts of questions may have unique answers; but for more complicated ones, however, especially hierarchical models, there may be large families of solutions. For example, to scratch your ear, you can move your hand to your ear, but when it gets there, your elbow can be in any of a number of different positions (close to your body or stuck out sideways). Thus, the motions of your upper and lower arm are not completely specified by “scratch your ear.” Solving inverse kinematic problems can therefore be difficult; however, by reducing the number of solutions by applying constraints to the problem (for example, “make the potential energy of your arm as small as possible”) the solution may become unique. In reality, objects are constrained by the way they are constructed and their environment.

Inverse kinematics has been applied to the animation of articulated human figures [Badler et al. 1987; Girard and Maciejewski 1985]. Furthermore, the systems of equations arising from solving these inverse kinematic problems are typically solved by iterative numerical techniques, and may take a long time to converge.
2.4.3 Dynamics

_Dynamics_ takes into account the physical laws that govern kinematics (for example, Newton's laws of motion, or the Euler–Lagrange equations). In the case of describing the forward dynamics of an object with Newton's second law of motion — force equals mass times acceleration — the mass remains constant, the force is known, and acceleration can be computed. A dynamic description of this scene might be, "at time \( t = 0 \) a cube is at position \((0, 100, 0)\), the mass of the cube is 100, and the force of gravity acts on the cube." The dynamic simulation of this scene, results in a falling cube.

Like kinematics, the dynamic description of motion can be inverted, and the dynamic problem becomes, "what force must we apply to the cube to get it to \((12, 12, 42)\) in 5 seconds?" This type of problem has received wide attention especially in the animation of articulated human figures [Armstrong et al. 1986; Wilhelms 1987].

The systems of equations arising from such inverse problems are typically solved by iterative numerical techniques. As in kinematics, constraints have been studied [Barr and Barzel 1988]; however, constraints for a dynamic model may be much more complex. To simulate the dynamic behavior of a system, _dynamic constraints_ are used. Forces are adjusted to act on an object so as either to achieve or to maintain some condition. For example, the force a floor exerts on the bottom of your foot is just enough to prevent you from penetrating the floor; however, this opposing force will dynamically change as the foot is in motion. Baraff [Baraff 1989] has extensively studied the motion of non-penetrating rigid bodies with dynamic constraints in mind.

2.4.4 Optimal Control

In some cases, an object's positions in the course of an animation are completely determined by physics, and computing these positions may be difficult. To facilitate the automation of this process, an animation technique for determining the intermediate positions
between an initial and final frame has been developed [Witkin and Kass 1988]. Witkin and Kass have used the term *spacetime constraints* to describe their optimal control technique.

In this method, the parts of the model are viewed as a function in time changing states (positions and velocities), where each time step is associated with a collection of numbers describing the state of the assembly. Thus, the function can be thought of as describing the path of the model. However, the possibility of multiple paths describing the motion exist. Some paths will have a total energy expenditure lower than that of others, or represent physically possible sequences of events (for example, the momentum of each part is proportional to the derivative of the part’s position), and there are also some paths whose initial position for the parts is the desired initial position and whose ending position is the desired ending position. The measurement of how far a path is from satisfying the problem’s constraints forms the basis for determining the optimal path.

The optimal path is computed by using *variational calculus* for finding minima of ordinary real–valued functions. An initial path is chosen, and altered slightly by moving certain points on the path in some direction. The path is then evaluated to determine if it is closer to a good path (where “good” means “low energy expenditure,” “laws of physics satisfied,” and “starting and ending conditions satisfied”) or is farther away. If it is closer, it becomes the new path, and the search is repeated. If it is farther away, the original path is altered by exactly the opposite perturbations, and this becomes the new path. As the process is iterated, a low–energy path that satisfies the constraints is found.

Motion simulation through optimization has also been used to model an object’s motion with differential equations [Brotman and Netravali 1988], and to build state–space controllers to provide control torques for cyclic movement like walking [van de Panne, Fiume, and Varanesic 1990]. Further related approaches to the spacetime constraint paradigm are reported in [Cohen 1992; Ngo and Marks 1993; Liu et al. 1994].

The spacetime constraints formulation is a powerful paradigm for indirectly controlling
the physically realistic motion of articulated figures. The essence of this approach is to have an animator specify what the animated object should do, and the computer determines how to do it. From the perspective of this research, where an object’s path follows cartilage morphology, spacetime constraints provides an interesting basis for computing a joint’s range of motion. In Chapter 4 this method has been pursued, and a new variation of this technique is presented.
Chapter 3

Modeling

In this work, the subtalar joint is reconstructed from scanned medical data into a computer graphics representation through a process termed volume modeling. This method is comprised of three steps: data acquisition, pre-processing, and segmentation.

Before discussing these activities let us examine the anatomy of the object we desire to model. The subtalar joint complex is formed by two ankle bones termed the calcaneus and talus. The articular cartilage separating these bones is also of interest because it facilitates the joint’s basic motion of eversion (turning outward) and inversion (turning inward) [Sarrafian 1993]. These motions are important because they are a significant aspect of walking, running, or jumping [Kapandji 1987]. In Figure 1 a side view of the subtalar joint’s bone and cartilage structures is shown. The calcaneus is the largest bone in the foot.
It is an irregular rectangular solid with three articular facets supporting the body of the talus. The talus has a dome shape and resides above the calcaneus but below the tibia and fibula of the lower leg.

The first step toward modeling the subtalar joint is to choose a medical imaging technique suitable to modeling the bone and cartilage structures.

### 3.1 Data Acquisition

The last twenty years have seen the advent of medical imaging systems that can non-invasively generate accurate volume information of living patients. These technologies include computed tomography, magnetic resonance imaging, and nuclear medicine modalities such as positron emission tomography and single photon emission computed tomography. Of these imaging techniques, data has been acquired using magnetic resonance because it is the best for identifying cartilage [Roberts and Pomeranz 1994]. It is the newest advance in musculoskeletal imaging and is able to discern subtle contrast differences between soft tissue structures such as: colon, fat, muscle, nerve, tendon, ligament, blood vessels, and cartilage [Jacobs 1989].

A magnetic resonance scanner outputs a series of slices where each slice represents a thickness of two to ten millimeters and typically contains 256 by 256 pixels. A value is stored at each pixel representing various characteristics about the molecular emissions of tissue subjected to a high intensity magnetic field and irradiated with pulses of low-level microwave radiation. By modifying the frequency and timing characteristics of the excitation pulse, it is possible to image particular types of molecules (for example, water). In this case, water and fat have a large value (shown bright in this chapter’s figures), bone has a small value (dark), and cartilage has an intermediate value (grey). The slices can be concatenated into a volume representation. In Figure 2 three orthogonal slices of the subtalar joint from magnetic resonance data are shown.
There are problems associated with this imaging technique. The output values at each pixel are not calibrated to any particular scale. The values will vary depending on the scan parameters and the patient's size and magnetic characteristics. Additionally, the data values of each pixel are not constant over the entire scan space — inhomogeneity in the magnetic field cause data values to be different for voxels that represent the same tissue. This problem has caused particular difficulties during the segmentation process (Section 3.3) of dividing the medical data into coherent regions of the same image property. The solution is to use interactive segmentation techniques that can make use of an expert's knowledge of anatomy and radiology.
3.2 Pre-processing

The purpose of the pre-processing stage is to perform any image processing before segmentation. The three common operations are: data loading, reformatting, and resampling.

3.2.1 Data loading

The purpose of this task is to load data acquired from the medical scanner into a computer graphics representation. Medical data from acquisition typically has a fixed layout consisting of headers and uncompressed image data. The headers contain information relevant to the study, including the patient’s name, scan date, resolution, and size (a metric measurement) of the volume scanned. The headers are skipped and the image data is stored as a three dimensional array of scalar numbers where each number represents magnetic characteristics of the sampled tissue. The size of the volume scanned is useful to associate with a computer graphics representation because the aspect (a ratio of a voxel’s width to its height to its depth) can be used for accurate modeling and volume rendering. Typically, the aspect is non-uniform and resampling techniques (Section 3.2.3) are applied to make the aspect ratio uniform.

3.2.2 Reformatting

Reformatting is a process to change volume data organization through fixed 90 degree rotations. The original organization of the medical data will depend on the scanned patient’s position and on the processing algorithms applied during acquisition. Thus, the sagittal plane (a cross section from the side of the foot) could be along the X–Y, X–Z, or Y–Z axes of the volume data set. This is undesirable because motion simulation requires the constructed model to have a specific organization — the sagittal plane must be aligned with the Y–Z axes of the volume data. Volume reformatting can be used to overcome this problem.

In Figure 3 several examples of reformatting the volume data set of Figure 2 are shown.
Figure 3: Reformatting medical data of the subtalar joint

The original image has been extracted from the X–Y plane (a sagittal view of the subtalar joint). In each subsequent image the volume data has been rotated and the X–Y plane has been reprojected.

Medical literature has resolved some of the ambiguity in discussing radiographic images. The terms *axial*, *coronal*, and *sagittal* refer to a specific cross section of the human body. If a human were standing erect in the Y axis and facing toward the Z axis of a left handed Cartesian coordinate system; the X–Z, X–Y, and Y–Z axes would refer to the axial, coronal, and sagittal cross sections of the human body respectively.
3.2.3 Resampling

Resampling is a technique used to convert a volume's aspect ratio from non-uniform to uniform. A non-uniform aspect implies that the voxels are non-cubic. They have a rectangular shape but the voxel height is not the same as its width or depth measurements. The motion technique described in this thesis requires cubic voxels for computing volume of intersection. The solution is to resample the volume data set at a new resolution such that the voxels are cubic. First, a new resolution is determined based on the size of the smallest dimension (for example, the height of each voxel). Second, the volume is resampled at a higher resolution such that all voxels are cubic. Third, the new value is determined using tri-linear interpolation between eight sample points of the original volume. The new value is determined by evaluating the equation:

\[Ax + By + Cz + Dxy + Eyz + Fxz + Gxyz + H = f(x, y, z)\]

where the variables \(x\), \(y\), and \(z\) represent the voxel's normalized local coordinate bounded by the eight data values \(A, B, C, D, E, F, G, \) and \(H\) of the original volume. Tri-linear interpolation is not a new technique, and has often been referred to in the visualization literature [Höhne and Bernstein 1986; Cline et al. 1988; Levoy 1988; Upson and Keeler 1988].

Typically, medically scanned data has less resolution in one dimension, and additional slices are created after the resampling process. In Table 1 interpolation has been applied to a magnetic resonance data set of the subtalar joint, and the new resolution and size are shown. As expected, the resolution in the third dimension increases and the size of the volume remains constant.

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution (number of elements)</td>
<td>(256, 256, 60)</td>
</tr>
<tr>
<td>Size (millimeters)</td>
<td>(160, 160, 63)</td>
</tr>
</tbody>
</table>

Table 1: Results after resampling
### 3.3 Segmentation

Segmentation is an essential step of the visualization process. It includes the partitioning of an image into sub-parts or objects, the grouping of those elements into logical collections, and the assignment of meaningful labels to those groups. Segmentation is used to partition the pre-processed volume data set into bone and cartilage structures for range of motion simulation. Segmentation and pre-processing both use a three dimensional array of scalar numbers; however, segmentation simply encodes voxels according to whether they are occupied by the structure of interest or not. This procedure is shown in Figure 4 where a two dimensional data set has been segmented into two images, and a reconstruction of the structures is shown. In this work four models (the calcaneus bone, calcaneus cartilage, talus bone, and talus cartilage) are independently constructed following a similar process.

Historically, segmentation has been approached from several directions, and a brief survey of the area is presented in Section 3.3.1. Based on these previous approaches, algorithms for segmentation have been developed and are discussed in Section 3.3.2. These algorithms rely on user interaction with a two dimensional display, and in Section 3.3.3, several display techniques are presented which have been implemented to assist segmentation.

#### 3.3.1 Background

There are several mechanisms that are used to perform segmentation ranging from the trivial and crude to the very complex and obscure. For the most part, only the simplest methods are in widespread clinical use today [Yoo 1993]. These methods are interactive and are loosely based on the paint program concept of colouring pixels, but in this case, voxels are classified according to the anatomical object they belong to. These “paint” methods are advantageous because they are robust (there is very little that can go wrong with a pixel painting tool) and it is not difficult to understand what they are accomplishing. However, they have the disadvantage of requiring a substantial amount of user time and
Two other approaches aimed at reducing the amount of user input and expertise required have been published in medical visualization literature. In the automatic approach, user input is not required and semantic or outside meaning is applied to assist segmentation. At present, these techniques have emerged only for very specific tasks and data; there are no
methods in sight that can handle arbitrary medical scenes [Kaufman et al. 1994] and the procedures are prone to unacceptable errors [Hanrahan 1993].

In the semi-automatic approach, a minimal amount of user input is employed. Typically, edge detection, region growing, and statistical pattern recognition are used for segmentation [Liang 1993]. A nice example is the volume seedlings approach by [Cohen et al. 1992]. They have developed an algorithm that allows the user to guide a region growing process. Instead of growing the region and displaying the final result, their system continuously highlights the region as it grows. The advantage of this kind of system is that it is much easier to catch segmentation errors before they occur.

An automatic algorithm would be the ideal approach to use; but if the segmented model is inaccurate, results from range of motion studies are meaningless. In comparison, semi-automatic and interactive approaches are as accurate as a user’s ability to identify bone and cartilage. In this work, the user is assumed to be an expert; thus, these techniques are reasonable to pursue as a segmentation method.

3.3.2 Algorithms

The algorithms chosen for segmentation are classified according to their behavior as: painting, region growing, region growing with distance, and region growing with a threshold. The user directs segmentation by selecting a voxel from a two dimensional display of the pre-processed volume data. A region is then grown from the seedling by marking voxels based on the algorithm used.

Painting

Painting is an interactive algorithm that enables the user to specify exactly which voxels to include in the segmentation. A region is grown from a user specified seedling voxel such that all neighbors within a specified radius are marked. The condition for including a
Voxel is mathematically described as:

\[ ||X - S|| \leq r \]

where \( X \) represents the vector position of any voxel in the volume, \( S \) represents the vector position of the seedling voxel, and \( r \) represents a user specified radius.

Voxel painting is a useful tool to interactively identify structures slice—by—slice and pixel—by—pixel. A radius parameter allows for different sizes of “brushes,” but the process is still labour intensive.

**Region growing**

In this work, a semi—automatic segmentation algorithm based on the use of region growing with statistical pattern recognition has been designed. The algorithm assumes the voxel values in the structure of interest are normally distributed and any new voxels added to the region must have their value within the mathematical condition expressed as:

\[ \bar{x} - s\delta \leq x \leq \bar{x} + s\delta \]

where \( \bar{x} \) is the mean, \( \delta \) is the standard deviation, \( x \) is the value of the voxel, and \( s \) is a user controlled parameter to manipulate the shape of the distribution. The mean of the voxel values in the distribution can be computed as:

\[ \bar{x} = \frac{\sum_{i=0}^{n} x_i}{n} \]

where \( n \) is the number of elements, and \( x_i \) is the value of the \( i \)–th element. The standard deviation can then be calculated as:

\[ \delta = \sqrt{\frac{\sum_{i=0}^{n} (x_i - \bar{x})^2}{n-1}} \]

The user directs the segmentation by specifying the deviation and selecting a seedling voxel. If the condition shown above evaluates to true, the voxel is added to the region and
Figure 5: Segmentation using region growing

the procedure is recursively continued with the four neighboring voxels. The region growing algorithm has been applied to segmenting the calcaneus bone shown in Figure 5. The segmentation is not entirely successful because several voxels in the interior of the structure were not included. The deviation parameter can be increased to include these voxels; however, this could result in a "spill" into structures that are not part of the calcaneus bone.

Region growing with distance

In this work, an alternative semi-automatic segmentation technique has been designed as a hybrid of the painting and region growing algorithms because the painting algorithm lacks automatic boundary detection and the standard region growing algorithm can produce "holes" if the deviation parameter is too small or "spill" if the parameter is too large. These two algorithms have been combined as a solution to these problems. The result is an algorithm that attempts to add neighboring voxels that satisfy the normal distribution and limited distance conditions. The normal distribution condition facilitates automatic boundary detection and the distance condition restricts the effect of a "spill" to a local region. The results of applying the algorithm are shown in Figure 6 and can be compared with the results of the standard region growing algorithm. There are fewer "holes" in the segmentation and the upper left bound of the bone has been accurately detected. From this example, the algorithm appears successful; however, the same problems as the standard region
Figure 6: Segmentation using region growing with distance

growing process exist, but the results have a limited area of effect.

Region growing with a threshold

Another approach to segmentation is called thresholding. The condition for including a voxel in a region is made by comparing the voxel's numeric value against a fixed number called a threshold. In general, it is not possible to specify a single threshold that will be good for an entire image [Parker 1994]; however, through a combination of thresholding and region growing, a new semi-automatic technique is proposed that can segment data over portions of an entire image. The user directs the segmentation — as in the previous techniques — by selecting a voxel from the two dimensional image. The voxel and its neighbors are then added to the region depending on the result of evaluating the following mathematical condition:

\[ t_l \leq x \leq t_u \]

where \( t_l \) is the lower threshold bound, \( x \) is the voxel's value, and \( t_u \) is the upper threshold bound. The "hole" artifact exhibited by the other region growing algorithms is still a problem, and is shown in Figure 7. Furthermore, determining a lower and upper bound is a trial and error process that often results in the "spill" artifact.
3.3.3 Display

The two dimensional display is an important component of segmentation because it facilitates identifying a voxel's classification and provides direct feedback from segmentation. Several computer graphics techniques have been used including: colour maps, flat or smooth shading, image blending, colour space transforms, image magnification, and volume reformatted projections.

Colour maps

Typically, a visualization application uses a colour map to associate a colour with each value in the data set. The problem of creating and manipulating the map has been resolved by using spline curves or ramp waves to establish a relationship between data values and colours. Spline curves provide an interactive method to create or modify a colour map by manipulating control points of a spline. There is one spline for each colour space parameter (for example, red, green, or blue). In Figure 8 a colour map has been created using splines to specify colour parameter intensities over a range of data values from zero to one.

Alternatively, ramp waves can be used to non-interactively specify a colour map by six controls. There are two controls for each parameter of the colour space. The first control is the initial value of the colour space parameter, and the second is the rate of change applied.
to it over the map. The colour map smoothly changes and this mechanism has been used to construct colour maps for the magnetic resonance figures of this chapter.

**Shading**

After the data values have been associated with a colour, each voxel can be rendered to the display as a constant or a smooth shaded rectangle. Constant shading holds the colour across the entire rectangle whereas smooth shading interpolates the colour between the four vertices. A comparison of the two shading schemes is shown in Figure 9. Smooth shading is appropriate for presentation graphics, but through experience, constant shading facilities a more precise identification of structural boundaries during segmentation.
Image blending

The display associates two variables — a value (from the magnetic resonance data) and a classification (for example, the voxel is part of the calcaneus bone) — with each voxel. These attributes are mapped to independent colours that are blended together before rendering a voxel in the two dimensional display. A transparency value is associated with a voxel’s classification and the two colours are blended together using alpha channel compositing [Porter and Duff 1984]. Mathematically, the blending operation can be formulated as a linear combination of two normalized colours:

\[ R = B (1.0 - \alpha) + F \alpha \]

where \( R \) is the resulting colour, \( B \) is the background colour, \( \alpha \) is the transparency value, and \( F \) is the foreground colour. In Figure 10 several examples of blending are shown. Blending enables the user to visualize both the classification of a voxel and its data value simultaneously. This facility has proved valuable during segmentation, because the user can control a transparency parameter \( \alpha \) to decide the importance of visualizing either the colour of the classification or the colour of the voxel’s value. For example, if during segmentation, a user decides that a voxel’s classification is more important than the colour associated with a voxel’s data value; the transparency parameter \( \alpha \) for the group’s colour can be increased.

Colour space transformations

The segmentation tool simultaneously displays an image of the current slice and each of
the segmented structures (the calcaneus bone, calcaneus cartilage, talus bone, talus cartilage), but only one structure can be edited at a time. The display highlights the selected structure and it is easily differentiated from the other segmented objects. The highlight is achieved by transforming a voxel’s red, green, and blue colour to Smith’s hue, saturation, and value model [Smith 1978], the value component is then increased and the resulting colour is transformed back. The intensity of the selected structure can be modified to decide the importance of visualizing either the colour of the image or the highlight of the current region being edited. The results of this technique are shown in Figure 11.

**Image magnification**

Some of the segmented structures are very small (for example, two to five voxels in thickness), and the interactive classification of a voxel is difficult to perform unless the image is magnified. Magnification is facilitated by using linear interpolation to scale a user specified region to the size of the display. The state is stored in a stack that facilitates the nesting of one magnification within another; furthermore, the stack can also be used to restore the display to a previous state. The results of magnifying an image are shown in Figure 12.

**Volume reformatted projections**

Identification of a structure’s extents in the third dimension are difficult; therefore, like Orthotool [Ney et al. 1989], a facility for multi-planar reformattting is provided. This technique is similar to changing a volume’s organization (Section 3.2.2); however, the physical data organization is not changed. Slices are extracted from the orthogonal planes of the
Volume reformatted projections are valuable for segmentation. Through experience, the posterior (the back or heel of our foot) and anterior (the toes or front of our foot) bounds can easily be determined in the sagittal plane whereas the lateral (the left side of our left foot) and medial (the right side of our left foot) extents are better identified in the coronal plane.

### 3.4 Implementation

The modeling task described in this section is composed of several modules implemented using the Silicon Graphics Explorer system. *Explorer* is a state of the art data-flow visualization system and includes a variety of modules for two and three dimensional visualization. New modules have been developed using the C programming language with
Implementation begins at the pre-processing stage where data is first loaded and converted from the scanning device format into a *uniform lattice* (a three dimensional Explorer data representation). The data is then reformatted and resampled. These operations have been implemented as Explorer modules and have been combined into a map where the output of one processing unit becomes the input of the next. A *map* is a collection of modules, connected in sequence, which process numerical data to accomplish a task. In Figure 14 the pre-processing map is shown.

Segmentation is the next stage of the modeling process. It is an interactive task that reduces complex pre-processed data into a volume representation that spatially enumerates voxels according to the group they belong in. Several algorithms and display techniques have been discussed to assist the identification and segmentation of objects. These techniques have been developed into an Explorer module that accepts a uniform lattice and colour map as input. In Figure 15 a map is shown indicating that the pre-processed...
output is used as input into both colour map and segmentation modules.

The method of segmentation is first selected from a menu that offers four different algorithms to choice from. The user then directs the segmentation by interacting with a dialog of parameters and a two dimensional display of a slice from the pre-processed data set. The algorithm is activated by using the mouse to select a voxel within the current slice; the two dimensional neighbors of the voxel are then marked or unmarked (if the selected voxel is already marked) depending on the algorithm’s behavior. Segmentation is not automatically performed in the third dimension because the full effect of changing the status of those voxels cannot easily be determined — the outcome of a three dimensional segmentation would be difficult to predict, visualize, and correct.

The segmentation algorithms are similar to the tools (for example, line, circle, or polygon) provided by a traditional drawing program because they can be used in collaboration with one another. For example, a structure could first be segmented with the region growing algorithm and later “touched up” with the painting algorithm.
Joint motion synthesis has important applications to the fields of biomechanics and medicine, because computer simulation can provide a means to analyze and test mathematical models. One problem is that simulation becomes more complicated as the degrees of freedom (the number of independent position variables necessary to specify the state of a structure) of joint motion increase [Watt and Watt 1992]. Secondly, an important aspect of motion simulation is collision detection. This calculation is very time consuming and is often restricted to studying only simple objects (for example, convex shapes) [Thalmann and Thalmann 1991]. The subtalar joint is not a simple object and has six degrees of freedom — fortunately its principal motion can be described by one parameter and constraints can be used to determine another. In this section a new technique that simulates the motion of arbitrarily shaped volume objects is presented.

Historically motion simulation has been approached with key–frames, inverse kinematics, dynamics, and spacetime constraints with optimal control [Cohen 1991]. Biomechanical Joint Motion Synthesis most resembles the work of [Witkin and Kass 1988] where spacetime constraints are used to determine an object’s intermediate positions. Objects are modeled as a physical assembly with various muscles that move the object’s parts. The path of the object over time is then determined using variational calculus to alter the path in some direction, and evaluate whether the path is closer to a good path (where “good” means “low energy expenditure,” “laws of physics satisfied,” and “starting and ending conditions satisfied”) or is farther away. If the path is farther away, the original path is altered by exactly the opposite perturbations, and the search is continued to find the path that consumes the least energy.
Both this work and the work of Witkin and Kass, entail the numerical solution of an optimization problem. In spacetime constraints, a constrained optimization algorithm called Sequential Quadratic Programming [Gill et al. 1981] is used to solve sparse linear systems. In this work any optimization method could have been used. The primary difference between these numerical techniques being that more or less computational time will be required to locate a maximum. The Fletcher—Reeves [Bunday 1984] optimization method is selected in Section 4.3.1, and in this technique new search directions are determined by computing a gradient unlike Sequential Quadratic Programming, which resolves this problem by calculating a computationally expensive inverse Hessian.

The criterion for measuring a “good” path is first described in Section 4.1. Next in Section 4.2, the constraints to the optimization method are discussed. Section 4.3 presents the algorithmic details, and Section 4.4 outlines the implementation.

4.1 Hamilton’s Criteria

Typically, an optimization criterion is expressed as a numeric energy function, and the optimization process involves evaluating the function to find a minimum or maximum. From an in—vitro study, Hamilton [Hamilton et al. 1995] has reported that the subtalar joint has a high cartilage contact area throughout normal range of motion, and conversely, has a low cartilage contact area when outside normal range of motion. The hypothesis he developed is that a path of motion can be computed by finding the positions of large contact area. Hamilton has also suggested that unsmooth contact is undesirable in the path of motion, and deformation distance (the distance between contact and non—contact) should be included in the criterion to offset the result of an extra large contact area when unsmooth contact occurs. The energy function he proposes is:

\[ f(x) = \frac{A(x)}{D} \bigg|_{V = V_0} \]

where \( A(x) \) is the computed contact area for joint position \( x \), and \( D \) is the computed dis-
Figure 16: Energy function for computing path of motion

tance between the position of touch contact (when the volume of intersection $V$ is zero) and the position where the volume of intersection $V$ is equal to the desired volume of intersection $V_0$.

This function can also be used to evaluate contact between arbitrary two dimensional objects. In Figure 16, the contact attributes are labeled on two ovals (one vertically above the other). Based on Hamilton’s work, the motion of the unfilled object over the top surface of the filled object can be determined by evaluating the contact for each unfilled object position. Those positions where the contact evaluation is large and the volume of intersection is equal to $V_0$ are then used to predict range of motion.

Range of motion can be thought of as a voluminous entity — a collection of positions with a high contact evaluation and no connection between them. We are interested in studying a path through this volume. Many paths can exist, and in Section 4.2, this problem has been addressed by prescribing one degree of freedom as a step constraint. The second problem towards predicting path of motion is calculating the deformation distance $D$. If we have a joint position where $V$ is equal to $V_0$, then there are many positions of non-contact. In Section 4.2, a second constraint is introduced to resolve this ambiguity.
4.2 Constraints

A constraint is a rule used to pre–determine an object’s position, and in this section, two design decisions are introduced to pre–determine the vertical translation and one other degree of freedom (for example, horizontal translation) of the talar position. The vertical constraint has been made to simplify computation of the non–contact position, and the second constraint is used to define the local neighborhood of optimization. Both constraints reduce the search space; thus computational time for optimization will be less. In Figure 17 these constraints are labeled on a reconstruction of the subtalar joint.

4.2.1 The Step (Horizontal) Constraint

One degree of freedom will be prescribed to specific values determined by a looping mechanism. This is best explained by an example where the horizontal component of the talus is stepped over an interval\(^1\) and for each step (Figure 18), the horizontal component is fixed and an exhaustive search could then be used to evaluate \(f(x)\) at every unit of verti-
One limitation of this constraint is that only one position will be predicted on the path of motion for each step. This is not a problem for simulating the subtalar joint because its motion has primarily one degree of freedom [Kapandji 1987]. In Figure 19, this motion is hypothetically illustrated in a two dimensional cross section. The calcaneus is stationary and two hypothetical talar positions are shown.

4.2.2 The Vertical Constraint

The second constraint uses the vertical component of talar movement to achieve varying...
amounts of volume intersection. A positive translation implies less intersection whereas a negative translation implies more intersection. The vertical component is constrained by the volume of intersection — the volume of cartilage intersection must be equal to $V_0$ for each evaluation of $f(x)$ (Section 4.1).

In a simple application of using the horizontal and vertical constraints, we horizontally step the talus over an interval, and for each step, the horizontal component is fixed and an exhaustive search could then be used to evaluate $f(x)$ at every unit or degree of depth, roll, pitch, and yaw. But before evaluating Hamilton’s criteria — as in the case of using only the step constraint, the vertical translation is determined such that the volume of intersection is equal to $V_0$. Through successive translations (using the current horizontal, depth, roll, pitch, and yaw), this component of the talar position can be found. Similarly, this process can be used to find a vertical translation resulting in non—contact (to obtain a value for the variable $D$).

An obvious limitation of this method is that the geometry of the object must be such that one structure is vertically above the other. Furthermore, the vertical component cannot be an important aspect of the range of motion. Based on the geometry and mechanics of the subtalar joint, both of these limitations are not serious problems to predicting range of motion.

The algorithmic details of calculating a reasonable position $x$ (of horizontal, depth, roll, pitch, and yaw), estimating the volume of intersection $V_0$, and computing contact area $A(x)$ are discussed next.

### 4.3 Method

The simulation predicts the path of the talus in relation to the calcaneus by determining those positions such that Hamilton’s criteria (Section 4.1) is maximal. The talar positions
foreach horizontal {
    do {
        Step One:
        Use optimization to predict depth, roll, pitch, and yaw.

        Step Two:
        Use $V_0$ to compute vertical for the contact position.
        Compute vertical for the non-contact position.
        Compute deformation distance $D$.

        Step Three:
        Evaluate contact area $A(x)$.

        Evaluate $f(x) = A(x)/D$.
    } while (Optimization of $f(x)$ is not locally maximal);

    Add position $x$ to path of motion.
}

Figure 20: The algorithm for path of motion simulation

are predicted by an optimization algorithm and are subject to the constraints discussed in Section 4.2. The simulation is comprised of three steps, and in Figure 20 the algorithm is shown using a horizontal step constraint. First, the calcaneus is held stationary while the talus is transformed into a position determined by the optimization algorithm. Second, the vertical component of the talar position is computed by rigidly compressing the volumes (Section 4.3.2); and lastly, having satisfied all components of the talar position, the contact area is computed (Section 4.3.3). Hamilton's criteria is evaluated and the result is then used as input back into the optimization method, and a new position is determined. The remainder of this discussion will follow the three steps outlined above.

4.3.1 Optimization (Step One)

The requirements of the optimization method have been broken into two tasks:

1. Determine positions for evaluating Hamilton's criteria.
2. Decide when the local maximum occurs.
These steps will be presented following a discussion of some background to other optimization techniques.

**Background**

One simple optimization method is to search for a minimum along the direction of steepest descent. This method is intuitive, but is not recommended for complex optimization problems because it is often slow in practice. The gradient is a local property, and following it requires frequent changes of direction leading to an inefficient computing procedure.

One method to solve an optimization problem is to develop methods which are based on quadratic functions because — through Taylor series expansion — any function can be approximated by a quadratic function in the vicinity of the minimum; therefore, any optimization method based on procedures that work for quadratic functions should be successful with other functions.

Any optimization method could be used for motion simulation, but the Fletcher–Reeves has been chosen because it exploits the fact that for a quadratic function of \( n \) variables, \( n \) linear searches along mutually conjugate directions will locate the minimum, and this approach is expected to locate a minimum before other methods such as steepest descent [Bunday 1984].

**Algorithm**

The algorithm for Fletcher–Reeves optimization is shown in Figure 21. The first step is to determine a starting position \( x \) (of depth, roll, pitch, and yaw) to evaluate. Obviously, the neutral or imaged position will be close to the range of motion; thus for the first iteration, the position \( x \) can be initialized to zero (there is no depth translation or rotation) or specified by the user, and for subsequent iterations, the starting position has been chosen to be the solution from the previous search.
Set $x = 0$, set $d = \nabla f(x)$, and set $i = 1$.

while ($\|d\| > \text{accuracy}$) {
    Use quadratic interpolation to find $\lambda$ such that $f(x + \lambda d)$ is maximal.
    Set $x_{next} = x + \lambda d$.
    if ($i = 4$)
        Set $d = \nabla f(x_{next})$ and set $i = 1$.
    else
        Set $d = \nabla f(x_{next}) - \frac{\nabla f(x_{next})^2}{\nabla f(x)^2} d$ and set $i = i + 1$.
}

Output $x$.

Figure 21: The algorithm for Fletcher–Reeves optimization

The next step is to compute the gradient of the initial position based on evaluation of the Hamilton’s criteria $f(x)$ at user specified intervals $s_i$. The gradient is formulated as a vector:

$$\nabla f = \begin{bmatrix} g_1 & \cdots & g_n \end{bmatrix}$$

where $g_i$ is approximated by calculating the central difference surrounding a position $x$:

$$g_i = \frac{f(x + s_i) - f(x - s_i)}{2\|s_i\|}$$

where $s_i$ is a vector of the user specified interval $s_i$.

The next step is to determine $\lambda$ such that $f(x + \lambda d)$ is maximized. A one dimensional numeric technique named quadratic interpolation [Bunday 1984] has been used. The algorithm uses a few function values, at particular points, in order to approximate a function by a simple polynomial over a limited range of values. The procedure computes the function at three distinct points ($\alpha$, $\beta$, and $\gamma$) and uses their function values ($f_\alpha$, $f_\beta$, and $f_\gamma$) to
approximate \( f(x) \) by the quadratic:

\[
\Phi(x) = Ax^2 + Bx + C
\]

where \( A, B, \) and \( C \) are calculated from the equations:

\[
A\alpha^2 + B\alpha + C = f_{\alpha}
\]

\[
A\beta^2 + B\beta + C = f_{\beta}
\]

\[
A\gamma^2 + B\gamma + C = f_{\gamma}
\]

and are substituted into the quadratic polynomial \( \Phi(x) \). Following this approximation, a maximum is estimated from the quadratic equation. Two parameters control the computational procedure: \textit{step} is used to compute two distinct points from the start position, and \textit{accuracy} is used to determine when the solution is close enough to the maximum. This terminating condition is based on the difference between the coordinates of the best and most recent estimation.

The last detail of the flowchart that needs to be mentioned is a statement of the conditions to completing the search. The algorithm will infer that a maximal solution is found if the length of the gradient is less than a threshold.

\textbf{4.3.2 Resolving Volume Constraints (Step Two)}

The primary purpose of resolving volume constraints is to determine the vertical displacement necessary to obtain a user specified volume of cartilage intersection between the talus and calcaneus. In Figure 22 this process is shown in two dimensions where two objects interact. In part A there is no vertical displacement and no intersection. The upper object is vertically displaced and an intersection is achieved in part B; however, the amount of intersection is larger than desired. Consequently, in part C, the vertical displacement and the amount of intersection are reduced. The secondary purpose of this task is to compute the distance \( D \) between the positions of contact (when the volume of intersection is equal to \( V_0 \)) and non-contact (when the volume of intersection is zero).
Figure 22: An example of vertical displacement to achieve intersection

```plaintext
function ContactSearch(desired volume) {
    do {
        Predict(vertical translation using binary or linear interpolation);
        Transform(talus by vertical translation);
        volume intersection = Intersect(talus with calcaneus);
    } while (desired volume does not equal volume intersection);
    return vertical;
}
vertical = ContactSearch(V_0);
D = contact – ContactSearch(smallest float point number);
Output vertical and D.
```

Figure 23: Algorithm to resolve volume constraints

In Figure 23, an algorithm to resolve these volume constraints is shown. The contact search is comprised of three steps. First, a one dimensional search is used to predict a vertical displacement that will achieve a volume of intersection equal to \( V_0 \) (for the position of contact) or zero (for the position of non-contact). Next, the volume is transformed, and lastly, the volume of intersection is computed. Each step is discussed separately.
Predict

The type of interpolation algorithm used depends on whether we are locating the contact or non-contact position. Linear interpolation is an effective prediction method for linearly increasing functions; however as in the case of locating the non-contact position, volume of intersection is constant for those positions when there is no contact. Therefore, linear interpolation has been used to predict the position of contact whereas a binary search has been chosen for locating the position of non-contact.

First, the prediction algorithm determines translations where the volume of intersection is less and greater than the desired amount. Second, a new position is predicted using a binary search or linear interpolation:

\[
\frac{x_n - x_p}{2} + x_p = \frac{V_0 - v(x_p)}{v(x_n) - v(x_p)} (x_n - x_p) + x_p
\]

where \(x_p\) is a vertical translation with a volume of intersection less than the desired, \(x_n\) is a position with a volume of intersection greater than the desired, and \(v(x)\) is the volume of intersection for position \(x\).

Transform

After predicting the vertical component of the talar position, the translation can be integrated with the five dimensional vector produced by the optimization step to create a complete description of the talar position. The talar model is transformed by multiplying each voxel’s coordinate by a transformation matrix. The matrix is obtained by first converting the talar position vector:

\[
[x \ y \ z \ roll \ pitch \ yaw]
\]

to a composition of primitive transformations:

\[
M_1 = R_z(roll) \cdot R_y(yaw) \cdot R_x(pitch) \cdot T(x, y, z)
\]

where \(R_z\), \(R_y\), \(R_x\), and \(T\) are the standard three dimensional transformation matrices described in [Foley et al. 1991].
The next detail to consider is the coordinate system of talar rotation. The transformation described above will produce an undesirable result of rotating the talus about the global coordinate axis of the subtalar joint. The solution is to translate the talus from its local origin \( \mathbf{x}_{\text{local}} \) to the global origin, apply the transformation, and then translate the object back. These transformations can be composited with the transformation matrix described above:

\[
M_2 = T(\mathbf{x}_{\text{local}}) \cdot M_1 \cdot T(-\mathbf{x}_{\text{local}})
\]

The origin of the talus \( \mathbf{x}_{\text{local}} \) is not clear — should the location be chosen by a user or calculated? The implementation supports both options. The origin can be specified by a three dimensional coordinate or calculated from the talar volume data. One method to calculate the origin is to use the physical concept of centre of mass, which refers to that point on an object that has the same amount of object around it in any direction. For example, a pencil’s centre of mass is the point at which the object can be balanced. In the case of the talus, its centre of mass is calculated from volume data by averaging occupied voxel coordinates along each dimension:

\[
C_x = \frac{\sum_{i=1}^{N} X_i}{N}, \quad C_y = \frac{\sum_{i=1}^{N} Y_i}{N}, \quad C_z = \frac{\sum_{i=1}^{N} Z_i}{N}
\]

where \( X_i, Y_i, Z_i \) are the horizontal, vertical, and depth coordinates of the \( i \)-th occupied voxel.

**Intersect**

Lastly, the volume of intersection between the calcaneus and talus is calculated. One requirement of computing this volume property is that small changes in translation should result in small changes in the volume measurement. A linear interpolation algorithm has been used to predict the position of contact; therefore, as one object intersects another we desire the volume of intersection to increase linearly.
An undesirable algorithm might sum the volume of each voxel in a discrete intersection (a volume where voxels are classified as completely intersected or not). This algorithm is not appropriate because small translations will produce a step in the number of intersected voxels, and a step in the volume measurement. In this case, the linear interpolation algorithm discussed previously would be ineffective at predicting the necessary displacement to achieve the desired volume of intersection $V_0$.

An initial approach is to estimate intersection as the distance between the centres of two voxels. Under various translations this measurement is reasonably accurate. However if one voxel is rotated, this method only approximates the volume of intersection. In fact, the worst case occurs when voxel centres are equal and one voxel is rotated by forty five degrees about two coordinate axes. The intersection is overestimated by 30%. This inaccuracy can be reduced by increasing the resolution of the volume model.

Voxel intersection cannot completely be estimated by the distance between two voxels. A transformed voxel’s centre is surrounded by at most eight neighboring occupied voxels of the static volume; thus, its intersection is related to its centre distance to each of the neighboring voxels. Mathematically, the voxel based intersection algorithm is then formulated as:

$$\sum_{i=1}^{n} \min_{j=1}^{8} p(||u_i - w_j||, 1.0)$$

where $u_i$ is the $i$–th voxel of the static volume, $w_j$ is the $j$–th surrounding voxel of the transformed volume, and $p(x)$ is a function to compute the percentage of intersection between two voxels given the distance (normalized to a number between 0 and 1 based on the voxel size) between them.

The derivation of the function $p(x)$ is not obvious, but if we examine a scenario of one voxel approaching a static volume and passing through it, we can make some generalizations about the desired behavior of this function. First, as the traveling voxel approaches
the perimeter of the volume we desire the amount of intersection to smoothly change; thus, the function \( p(x) \) must be at least \( C^1 \) continuous. Second, while the traveling voxel is passing through the volume, the volume of intersection should remain constant, and there should be one hundred percent intersection. Furthermore, this measurement should not fluctuate while there is total voxel intersection. Third, after the traveling voxel has passed through the volume, the percentage of intersection should drop to zero. These important properties of \( p(x) \) where a voxel intersects a volume are summarized below:

1. Voxel intersection over the volume exterior must be 0%.
2. Voxel intersection over the volume interior must be 100%.
3. Voxel intersection over the volume perimeter must be continuous.

The first condition can be resolved by evaluating \( p(x) \) only for those voxels with a centre distance less than a threshold (typically the voxel size). The second condition is more difficult to solve, but if we examine total intersection from a voxel’s perspective. We can assume total intersection if the voxel is surrounded by eight occupied voxels. Thus one requirement to formulating the function \( p(x) \) is that for any transformed voxel surrounded by eight occupied voxels, there is one hundred percent intersection. Mathematically, this condition is formulated as:

\[
\sum_{i=1}^{8} p(\|x - w_i\|) = 1.0
\]

where \( x \) is any position in the cube bounded by the vertices \( w_i \). Unfortunately, there is no function that satisfies this condition and the other constraints, but if we relax the expression to finding a function such that the percentage of intersection can be greater than or equal to one hundred percent, we can approximate the function by a linear, parabolic, or higher degree polynomial. The higher the degree, the more accurately this function can approximate the ideal function, and thus it can more accurately reflect the volume of intersection when there is not total intersection (for example, on the volume’s boundary). In this work, the ideal function has been approximated by a parabolic function defined geo-
From the discussion above, if the distance between two voxels is zero, there is total intersection. Thus the first coordinate (0, 1) must be on the parabolic curve. Next, the second coordinate (1, 0) has been obtained by observing that if the distance between two voxels is greater than or equal to their voxel size, there is no intersection. The third coordinate has been calculated by an analysis of a voxel placed between eight vertices such that two vertices are too far away to participate in the calculation, and in the worst case, the distance to four of the non-participating vertices will be equal to the voxel size. The percentage of intersection should then be distributed equally between the remaining two vertices. The distance $x$ to these two participating vertices can then be determined such that intersection is distributed equally (50%) to each vertex. The third coordinate on the parabolic curve has been computed as:

$$\sqrt{1 - \frac{\sqrt{2}}{2} \frac{1}{r^2}}$$

From these coordinates the percentage of intersection function $p(x)$ has been calculated, and the function has also been found to satisfy our third requirement. This continuous function is graphed in Figure 24.

**Evaluation**

One requirement of this algorithm is that the volume of intersection measure smoothly increases as perfect intersection is approached. It is also desirable to have an accurate
Table 2: Numeric results from volume intersection

This volume measurement has been tested in two examples over a variety of translations, and the estimations (Table 2) can be compared with analytical results. As the objects are rigidly compressed together the volume of intersection increases and smoothly changes. In both examples, two cubes with eight voxels of unit size have been modeled, and the upper cube has been vertically translated to obtain intersection. In example one (Figure 25) the upper cube has not been rotated, and in example two (Figure 26) the transformed cube has been rotated by forty five degrees along two axes (the worst case). The intersection algorithm has overestimated the analytical results by at most 17% and 27% in each example respectively; however, by increasing the resolution of the cube a more accurate estimation can be obtained.
An analysis of this measurement reveals that intersection can be overestimated when voxel centres are equal. The worst case occurs when one voxel is rotated by forty five degrees about two of its coordinate axes. Intersection will also be overestimated during perimeter intersection, and in this worst case there is no rotation between the voxels. Both of these inaccuracies can be reduced by increasing the volume’s resolution.

Even with the problems outlined above, this algorithm is acceptable for motion simulation because its use is insensitive to these inaccuracies. The algorithm is used to determine the volume of intersection between the calcaneus and talus under various vertical translations. One requirement for this application, is that the volume of intersection increases as the exact analytical measure increases. This behavior is demonstrated by both of the examples
shown above. Second, this algorithm accurately measures complete intersection, and this maximum will only occur when there is perfect intersection. Lastly, this measurement has the continuous behavior which is necessary for linear interpolation.

This algorithm completes the final component of resolving volume constraints. Vertical translations for the talar positions of contact and non-contact can be determined. The difference between these positions is then used to define the variable $D$, and the talar position $x$ in which to evaluate contact area $A(x)$.

**4.3.3 Contact Evaluation (Step Three)**

After determining all components of an object’s position, the contact area is computed and used to numerically evaluate whether one position is more likely on the path of motion curve than another. The method of calculating this property will first be described in two dimensions where arc length is used instead of area, and pixels are used instead of voxels. The scenario is shown in Figure 27 where the area of contact is indicated by the cross-hatched region, and the bold line denotes the length of contact. In the diagram, two measurements are shown and the choice for comparing contact area does not matter as long as it’s consistent (for example, always measuring the contact by the length of the upper line).
The arc length is measured by taking advantage of the object’s discrete representation to estimate this property by calculating the length of the line made up of pixels bordering the contact. This description is not specific enough to describe an algorithm; consequently, the previous scenario has been rasterized (Figure 28) to facilitate a mathematical description of these pixels. Through examination of the intersection and perimeter data sets, we notice that the pixels of interest are those that are contained in both sets.

In three dimensions, these sets can also be determined, and used to distinguish whether a voxel is bordering the contact or not. The area can then be computed by polygonalizing each voxel and summing the area of its polygons. The remainder of this discussion describes these three operations.

**Intersection**

The algorithm to compute intersection is based on the concept of a voxel exhibiting a field influence from its centre such that any other voxel within its field is included in the intersection. This procedure need only test the closest occupied voxel — which can quickly be computed through the use of a spatial data structure [Samet 1990]. The field size can vary, but for this work, the intersection should include all voxels that contributed to the volume of intersection; thus, the field size has been chosen to be the same as the voxel size.
Figure 29: The twenty six, eighteen, and six neighborhood perimeters

Perimeter

The perimeter of a region consists of the set of occupied voxels that have at least one unoccupied neighbor. The occupancy status of a voxel is easily determined; however, the neighbors of a voxel depend on the kind of connectivity desired. In [Kaufman 1987] three kinds of neighbors have been identified. There are six direct (face) neighbors, twelve indirect (edge) neighbors, and eight remote (corner) neighbors. The connectivity can then be classified as: six connected (the direct neighbors), eighteen connected (the direct and indirect neighbors), or twenty six connected (the direct, indirect, and remote neighbors). The choice of which neighbors are tested is important because it affects the connectivity of the resulting volume. For example, if only the six face neighbors of the voxel are examined, the resulting volume will be twenty six connected.

In this work, the perimeter must be six connected. This decision is based on the examination of results obtained by polygonalizing (with a threshold of one) the perimeter of a sphere which has been modeled as a volume of occupied (a value of one) or unoccupied (a value of zero) voxels. In Figure 29, the perimeters were calculated by inspecting the twenty six, eighteen, and six neighbors of each voxel. Analysis of these results reveal that the eighteen and six connected perimeters have fewer polygons and less surface area. In fact, holes can even be observed in the triangulation of the last example. This artifact can be attributed to ambiguous polygonalization. This phenomena is shown below in two dimensions where diagonally opposite sides of a lattice are included in the perimeter set.
There are two equally appropriate ways to construct an iso-surface between the points:

![Diagram of two equally appropriate ways to construct an iso-surface]

**Polygonalization**

The purpose of this stage is to generate polygonal iso-surfaces for volume data where each voxel has been classified according to whether its contained in the intersection and perimeter of contact. Following iso-surface extraction, the area of each polygon is computed and used to estimate contact surface area.

The first step of this problem is generating the three dimensional iso-surface. Several variations of the algorithm exist in computer graphics literature [Wyvill *et al.* 1986; Lorensen and Cline 1987; Cline *et al.* 1988; Gallagher and Nagtegaal 1989]; however in this case, scalar values are assumed to be given at each point of a lattice. A particular level surface is then approximated by determining all intersections of the level surface with the edges of the lattice by linear interpolation.

Each cube in the lattice now has some number of edges marked according to whether the edge is inside or outside the iso-surface. The arrangement of these points are then classified into several cases. For each case, a choice is made of how to fill in the surface within the cube. Figure 30 shows two such cases where black circles indicate that the vertex is on the inside of the surface.

This technique extracts iso-surfaces from sampled data by approximating the surface in each cell by polygons whose vertices lie between sample points with values on opposite sides of a threshold. Unfortunately, when a cell face has an intersection point in each of its
four edges, the correct connection among intersection points becomes ambiguous. The solution is to connect the iso-surface points in one manner as the face of one voxel and in another manner as the face of the adjoining voxel. In other words, different triangulations must be used. These modifications require the addition of considerable complexity; therefore, a natural question to ask is how often do these configurations arise. In the work of [Wilhelms and Van Gelder 1990], four sample data sets were considered and the frequency of ambiguous occurrences were low. Furthermore, in the work of [Nielson and Hamann 1991] three other examples were examined, and similar results were observed. In Table 3, the frequencies of ambiguous cases based on cells containing an iso-surface are summarized. Ambiguous polygonalization is infrequent. Furthermore, a consistent connectivity between vertices is not essential when computing contact area; thus, the polygonalization algorithm developed may not construct topologically correct surfaces when presented with an ambiguous configuration.

A complete study of ambiguous polygonalization and its affect on predicting range of motion should be done in the future. This study should include determining the frequency of these events in the subtalar joint, and a comparison of area measurements between topologically correct and incorrect polygonalizations. It can be hypothesized that this occurrence is infrequent and has little affect on contact area measurement.
### Table 3: Frequency of ambiguous polygonalization

Following polygonalization, the contact area is estimated by summing the area of each polygon. In [Goldman 1991], a formula to calculate this measure is derived from Stokes Theorem:

$$\frac{1}{2} \left| N \cdot \left\{ \sum_k P_k \times P_{k+1} \right\} \right|$$

where $N$ is the normal, and $P_k$ are the vertices of a planar polygon.

**Evaluation**

One question to ask is whether volume polygonalization is appropriate for measuring contact area? Over 250 successful clinical studies have been carried out to determine the medical significance of examining anatomy by polygonalization [Cline et al. 1988]. Through these studies, polygonalization has proven to be an acceptable method for modeling the anatomy, thus it should also be appropriate for calculating measurements like surface or contact area.

In this section, the intersect, perimeter, and polygonalization algorithms described previ-
Figure 31: Four positions of interest for ContactArea example one

Table 4: Results for ContactArea example one

<table>
<thead>
<tr>
<th>Resolution</th>
<th>No Contact</th>
<th>Touch Contact</th>
<th>Half Intersection</th>
<th>Full Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exact</td>
<td>Estimate</td>
<td>Exact</td>
<td>Estimate</td>
</tr>
<tr>
<td>2 x 2 x 2</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>3 x 3 x 3</td>
<td>0.0</td>
<td>0.0</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>4 x 4 x 4</td>
<td>0.0</td>
<td>0.0</td>
<td>9.0</td>
<td>9.0</td>
</tr>
<tr>
<td>10 x 10 x 10</td>
<td>0.0</td>
<td>0.0</td>
<td>81.0</td>
<td>81.0</td>
</tr>
</tbody>
</table>

Table 4: Results for ContactArea example one

ously have been combined to compute contact area, and have been evaluated by two manu-
factured examples. In the first example, the contact area is calculated between two cubes. Using these models, four positions of interest have been identified, and are shown in Figure 31. These positions have been named: no contact, touch contact, half intersection, and full intersection. The model is composed of a discrete number of unit sized voxels to satisfy a particular resolution. In Table 4, the results have been tabulated using an iso–surface and intersection threshold of one. The difference between the expected and computed solutions for the half intersection can be attributed to the intersection algorithm. There are no partial intersections. Thus, when half intersecting two cubes with a resolution of four, the resulting resolution is 4 x 3 x 4.
In Figure 32, the second example is shown. The sphere has a diameter of seven and the cube has a length of nine along each dimension; thus, the cube encompasses the sphere when their centres coincide. Contact areas over both the sphere and cube have been calculated for different relative positions. In Table 5, the results are shown using the same parameters as in the first example. The volume model poorly approximates the sphere; consequently, the sphere is modeled as several flat faces, and the difference between exact and estimated results can be explained by this phenomena.

The exact measurements in the table were calculated from an equation, whereas the esti-
mated results were calculated discretely using the algorithm discussed in this section. If the resolution of the sphere or cube is increased, the contact estimation can better approximate the analytical exact. Thus, in the case that the volume representation perfectly matches its modeled object (for example, in modeling a cube), the estimated measurement is the same as the exact (Table 4).

Following the computation of contact area $A(x)$, Hamilton's criteria $f(x)$ can be evaluated, and this measure can be used as input into the optimization process.

### 4.4 Implementation

The simulation has been implemented as a number of Explorer modules. The *Fletcher-Reeves* module uses an optimization technique to predict positions on the range of motion. These positions are constrained to achieve a constant volume of intersection $V_0$ by the *ContactVolume* module, and a third module named *ContactArea* evaluates the contact area for each position. The implementation of these module is discussed in this section.

#### 4.4.1 FletcherReeves

The Fletcher–Reeves algorithm has been implemented as an Explorer module using the C programming language. The modular design has inhibited implementation because evaluation of Hamilton's criteria $f(x)$ requires leaving the module to pass information — the joint position to evaluate — to a subsequent module. This problem has been solved by using a state machine model to facilitate the recall of a state from a previous execution. When this function needs to be evaluated, the *FletcherReeves* module saves its state, outputs a position, and waits until results are obtained from both the *ContactVolume* module (the vertical translation, and deformation distance $D$), and the *ContactArea* module (the contact area). After executing the optimization procedure for each horizontal position, the range of motion is output as a one dimensional lattice of positions.
A number of parameters are provided that the user can use to manipulate the behavior of the algorithm (Figure 33). The horizontal dials (including Horizontal Step) control the number of horizontal positions that the range of motion will be computed for. The step parameters specify the central difference for computing the gradient, and the linear controls can be used to alter the behavior of the quadratic search algorithm. Lastly, the accuracy parameter affects the tolerance used for ending the search process.

In Figure 34, the module has been integrated into an Explorer map containing the ContactVolume and ContactArea modules. The FletcherReeves module behaves as a loop con-
troller for the other modules, and Trigger3 is used to synchronize the output from the ContactVolume and the ContactArea modules before starting the next loop. Notice that contact area can not be evaluated until the vertical component has been resolved by rigid compression of the talus onto the calcaneus.

4.4.2 ContactVolume

The ContactVolume stage is comprised of three modules that predict, transform, and compute a volume of intersection. The user can manipulate the behavior of this process by interacting with the widgets in the ContactVolume dialog (Figure 35). Two of the parameters, accuracy and maximum, affect conditions for terminating the search. The first controls accuracy of the computed translation (for example, an accuracy of twelve will guarantee the computed translation to be within twelve units of the exact solution), and the second parameters affects the maximum distance the algorithm will translate a volume to locate the initial values \( x_p \) and \( x_n \). The step parameter’s sign indicates the direction of rigid compression (for example, a negative step will compress the talus vertically downward),
and its magnitude denotes the locality of searching for the initial values \( x_p \) and \( x_n \). Lastly, the volume widget can be manipulated to alter the desired constant volume of intersection \( V_0 \).

The modules have been implemented using the C programming language and the Explorer module builder. The ContactVolume module, which performs the search component, behaves as a loop controller for the process and has been implemented using the state machine model. The three modules have been combined into an Explorer map and are shown in Figure 36. The loop controller accepts a vector position as input from an upstream module (for example, FletcherReeves) and outputs a transformation matrix. The Transform module applies the matrix to volume data (usually loaded from secondary storage) and outputs a transformed data set. The Intersect module then calculates the volume of intersection and outputs the result for input into ContactVolume. Once a solution is found, ContactVolume outputs the vertical translation, the deformation distance, and a transformation matrix for use by the ContactArea process.

### 4.4.3 ContactArea

The polygonalization and area calculations have been combined into an Explorer module that accepts three scalar uniform lattices: the input volume, the input’s perimeter, and the intersection of the input with the transformed volume. The iso–surface threshold is a parameter that can be controlled by interacting with the dial in the module’s dialog (Figure...
The contact area stage is divided into three tasks: intersection, perimeter, and polygonalization. Each module has been implemented using the C programming language and the Explorer module builder. These modules have been combined into an Explorer map (Figure 38).
Chapter 5

Results

In this section, results from running the simulation in two environments are discussed. First, the simulation will be examined under controlled conditions where ideal results can be predicted. Synthetic volume models of geometric shapes have been constructed and used to evaluate whether the optimization method is appropriate for determining positions where Hamilton’s criteria is maximal. The Fletcher-Reeves optimization method will be compared to an exhaustive linear search.

In the second environment, the calcaneus and talus cartilage have been modeled from medical data. The simulation has predicted subtalar joint motion by optimizing Hamilton’s criteria and contact area. Results are compared to my own empirical observations and relevant biomechanics literature.

5.1 Synthetic Data

The simulation has been tested by two synthetic examples. The goal is to evaluate whether optimization is a reasonable method to maximize Hamilton’s criteria. The optimized positions will be compared against an exhaustive linear search of all possible positions on the basis of similarity between positions and criteria evaluation. The exhaustive linear search constructs the path of motion by discretely sampling the search space and determining a maximum for each frame (horizontal step). This method is unlikely to find a maximum due to its discrete nature, and will not necessarily preserve continuity between frames (for example, from one frame to the next, some components of the position could radically change). The exhaustive search remains an acceptable control model because its accuracy
is bounded by the sampling interval, and discontinuity is reduced by carefully choosing the sampling parameters and synthetic examples.

In the first example, a cube is positioned vertically above a sphere. The cube is then translated horizontally and the simulation predicts positions of maximal contact. The cube is expected to rotate about the depth axis (yaw). In Figure 39 three frames from the simulation are shown exhibiting this characteristic. The sphere is modeled using a volume representation; consequently, the sphere is not smooth and we can expect small deviations in the depth, roll (vertical axis rotation), and pitch (horizontal axis rotation). In Figure 40 the optimized positions are shown graphed against the horizontal translation. The most interesting aspect of this graph is the stair case in yaw. This phenomenon can be explained by examining the volume representation of the sphere. Its surface is faceted, thus there are abrupt changes in the cube’s rotation to maintain contact. The brief increase in rotation before each step can also be rationalized in this manner — there is a small plateau of voxels before each change in facet (Figure 41).

An exhaustive linear search has been performed using the sampling parameters shown in Table 6, and in Figure 42 maximal positions are shown graphed against the horizontal translation. A comparison between the exhaustive search and optimization shows that there are many aspects of similarity. In particular, both graphs exhibit a stair case in yaw rotation. This simulation is symmetrical about depth, roll, and pitch (for example, a roll of $-5^\circ$ will have the same contact evaluation as a roll of $5^\circ$). Thus, it is irrelevant to weight
Figure 40: Graph of optimized positions for synthetic example one

the sign when comparing symmetrical positions. The optimized depth, roll, and pitch are comparable to results obtained by exhaustive search. Lastly, before discussing the next synthetic example, the optimization method can be seen preserving a higher degree of continuity between frames.

In the second example, a more complex scenario is presented. A cube with a track for an ellipsoid has been modeled. The width of the track and ellipsoid are 11 and 9 respec-

Figure 41: Volume representation of the sphere
Table 6: Exhaustive sampling interval for synthetic example one

<table>
<thead>
<tr>
<th>Degree of Freedom</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Number of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth</td>
<td>-1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Roll</td>
<td>-5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Pitch</td>
<td>-5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Yaw</td>
<td>-70</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 42: Graph of linear search positions for synthetic example one

respectively, thus there is freedom for limited translation and rotation. The motion within the track is simulated by horizontally translating the ellipsoid and determining positions where Hamilton’s criteria is maximal. The yaw is expected to increase then decrease as the ellipsoid leaves the track and contacts the planar surface. In Figure 43 three frames from the simulation are shown.

The simulated motion of the ellipsoid is graphed in Figure 44, and the ellipsoid’s coordinate system is labeled in Figure 45. The graph confirms the expected behavior of yaw
rotation, but there are also small changes in depth, roll, and pitch. These actions can be explained by examining the polygonalization and contact intersection between the cube and ellipsoid. Three images from the simulation’s fourth frame are shown in Figure 46 to assist analysis. The left and centre images display the polygonization of the cube with the ellipsoid, and the right image presents the intersection of these objects. The roll can be attributed to the optimization method attempting to maximize contact with the left, right, and front sides of the track. There is also a slight increase in the pitch which can be explained in that a rotation enables the ellipsoid’s left facet to better match the shape of the track. With a rotation, a criteria value of 19.77 has been measured, and in comparison, a
Figure 45: Coordinate system of the ellipsoid

criteria value of 19.54 has been measured with no rotation (all other components of the position are fixed). The graph also shows an increase in depth translation. This phenomenon is justified because as the ellipsoid leaves the track, its vertical translation increases, and the bottom centre voxels become contact voxels. The ellipsoid has been translated such that cube voxels left and right of the ellipsoid’s centre both intersect the ellipsoid and the contact is then larger.

The last component of the synthetic evaluation is to compare the criteria measurements between the optimization and exhaustive linear search. The sampling parameters of the exhaustive search are shown in Table 7, and in Figure 47 the measurements are graphed

Figure 46: Polygonization of the fourth frame
Table 7: Exhaustive sampling interval for synthetic example two

<table>
<thead>
<tr>
<th>Degree of Freedom</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Number of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Roll</td>
<td>-5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Pitch</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Yaw</td>
<td>0</td>
<td>15</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 47: Graph of Hamilton's criteria for synthetic example two

against the ellipsoid’s horizontal translation. There are many similarities between the graphs. In particular, the optimization method closely follows the exhaustive search until the tenth horizontal translation. The exhaustive search deviates from the optimization method, which may be attributed to its lack of continuity between frames. However, it is interesting to note that the solutions later converge.

The optimization method has been successfully tested by two synthetic examples. The positions and measurements of Hamilton’s criteria have been correlated with results obtained from an exhaustive linear search. We will now evaluate the simulation of subtalar
5.2 Medical Data

The simulation has been tested using scanned medical data. The objective of this discussion is evaluate whether Hamilton’s criteria and its implementation are reasonable for simulating joint motion. The results will be compared with my own empirical observations and relevant biomechanics literature that describes the average motion of the subtalar joint. In addition, Hamilton’s criteria (contact area divided by deformation distance) we will compared to a solution obtained using only contact area. I expect the simulated joint positions and maximal contact measures to have a well defined continuous pattern (for example, a linear increase or parabolic change of the measured value as a function).

In Section 5.2.1, the typical motion of the subtalar joint is presented. Next in Section 5.2.2, the subtalar joint is simulated using a horizontal step constraint; however, the results are not in agreement with biomechanical studies. The simulation has since been re-examined, and through a hands on experience with cadavers and bone skeletons, it has been determined that roll would be better suited as a step constraint. Unfortunately, in Section 5.2.3 unsuccessful results are also reported for this. Lastly in Section 5.2.4, the parameters used for producing these results are discussed.

5.2.1 The biomechanical motion of the subtalar joint

The patterns in talus movement have been observed through an empirical study. The talus bone of a skeleton was manually stepped in the horizontal and roll directions. The observed movement in the other degrees of freedom are tabulated in Table 8.

The typical motion of the subtalar joint has also been reported in biomechanics literature. Quain [Quain 1892], Peirsol [Piersol 1923], and Cunningham [Cunningham 1927] state that the primary motion of the subtalar joint is a rotation about an oblique axis. A more
<table>
<thead>
<tr>
<th>Horizontal</th>
<th>Depth</th>
<th>Roll</th>
<th>Pitch</th>
<th>Yaw</th>
</tr>
</thead>
<tbody>
<tr>
<td>increase</td>
<td>decrease</td>
<td>decrease</td>
<td>undecided</td>
<td>decrease</td>
</tr>
<tr>
<td>undecided</td>
<td>undecided</td>
<td>increase</td>
<td>decrease</td>
<td>decrease</td>
</tr>
</tbody>
</table>

**Table 8: Empirical observations of subtalar joint movement through a horizontal and roll step**

Precise definition of the axis of the subtalar joint was achieved by Manter [Manter 1941] using available technology and largely visual measurements of instrumented cadaver specimens. In his 16 specimens, he noted the inclination of the axis of the subtalar joint to be 42° (range 29 to 47°) as measured upward from the horizontal plane and projected on the sagittal (vertical depth) plane, and deviating 16° (range 8 to 24°) medially from a line connecting the midpoint of the heel to the first web space, then projected onto the transverse (horizontal depth) plane. But even among his 16 specimens he found considerable variation.

Isman and Inman [Isman and Inman 1968], using a similar technique measured 46 cadaver feet. Their average angles were very similar — 42° vertical and 23° medial — with a slightly larger medial deviation being accounted for by their use of the midline of the foot rather than the first web space as a landmark. These authors also noted substantial individual variation. In their series, range of measurements was 20 to 68° in the sagittal plane and 4 to 47° of medial deviation (transverse plane).

The subtalar axis of rotation has been reproduced from [Grayson 1991] with permission in Figure 48 and Figure 49. The simulated motion of the subtalar joint will be compared to these biomechanical results.

**5.2.2 Simulating subtalar motion by horizontal steps**

In Figure 50, the talus cartilage in relation to the calcaneus is shown on the left, and its coordinate system on the right. This figure should be referred to during the course of
Figure 48: Variations in position of axis as projected upon sagittal plane

Figure 49: Variations in position of axis as projected upon transverse plane

examining the graphs of this section. The joint has been simulated using Hamilton’s criteria, and the optimized positions are graphed against the horizontal translation in Figure 51. Brief discontinuities are shown, but the positions exhibit a linear pattern. The depth translation decreases as the horizontal translation increases. This is an acceptable pattern given the joint’s geometry shown in Figure 50. However, it is difficult to determine an expected
behavior for roll, pitch, and yaw. Thus, cadavers and bone skeletons have been examined. The graphed patterns of roll and yaw match my own empirical observations; however, the pitch should not have significantly changed. I suspect the simulation obtained a larger sur-
face area of contact by maintaining simultaneous contact between anterior (front) and posterior (rear) facets during positive horizontal translation. These positions are probably outside of the normal path of motion. After all, horizontal translation is not a significant component of subtalar joint motion.

In Figure 52, the joint positions obtained by optimizing contact area are graphed, and similar patterns are observed.

The maximal measurements from the two previous simulations are graphed in Figure 53 and Figure 54 respectively. Hamilton believes his measurement should drop as the limits of joint motion are reached, thus the sudden drop in contact measurement at −5 and 6 maybe explained by this. However, we should identify why the measurement then significantly increases from −5 to −7. These positions may be outside of the joint’s range of motion, but it may also be an indiction that deformation distance has a noise like behavior.
In my opinion, it seems reasonable that one position — near the neutral position — would have a contact measure greater than any other position in the path of motion. In other words, the other positions would not be as congruent. This behavior is exhibited by the
The results from stepping horizontally are not in agreement with biomechanics literature. The roll and yaw should be far more pronounced. The axis of rotation is not even close to the studies discussed in Section 5.2.1. The best results were obtained using Hamilton’s criteria where angles from the horizontal plane and midline are estimated as $-42^\circ$ and $-36^\circ$ respectively. These poor results may be attributed to using horizontal translation as a step constraint. The primary motion of the subtalar joint is rotation. Roll significantly changes throughout joint motion, thus it has been chosen as the new step constraint. Results obtained are discussed next.

### 5.2.3 Simulating subtalar motion by roll steps

The roll has been used as a step constraint, and the other degrees of freedom (horizontal, depth, pitch, and yaw) are solved by optimization. In Figure 55 the positions obtained by optimizing Hamilton’s criteria are graphed. The pattern in yaw is in agreement with my
own empirical observations — the yaw decreases as roll increases. The horizontal and depth translation do not vary much, but this is expected because the talar centre of mass is located near the joint’s axis of rotation (Figure 56). Thus, while the joint is rotating there will not be substantial variation in these translations. In the graph, the pitch increases as roll decreases, but it is not clear whether its magnitude is correct. The pitch should slightly increase. I am not sure if this discrepancy is enough to affect a correlation of results to biomechanics literature.

Results have also been obtained through optimizing contact area. In Figure 57 positions are graphed against roll, and similar patterns to the previous graph are observed in the horizontal, depth, and pitch. However, empirical observations and the previous results suggest that the yaw should decrease more significantly as the roll increases.

In both graphs there are small discontinuities near the neutral position. This can be attributed to an artifact of the volume of intersection and contact area measures. When the volume data sets are aligned contact area is overestimated, volume of intersection is underestimated, and the optimization method has difficulty escaping a local maximum. The volumes later become unaligned as we step in roll, and this problem is not as significant.

The contact measurements of Hamilton’s criteria and contact area are graphed in Figure 58 and Figure 59 respectively. A discontinuity is observed at the neutral position; other-
Figure 57: Graph of positions obtained by stepping roll and optimizing contact area.

Figure 58: Graph of Hamilton's criteria by stepping roll.

wise, both graphs have a continuous behavior. In the future, a simulation should be done to determine the congruity at the limits of joint motion. In particular, we are interested in
whether the contact measurement suddenly drops — as Hamilton proposes — or some other behavior is exhibited.

Both simulations are not in agreement with biomechanics literature even though patterns are similar my own scientific observations. Hamilton’s criteria better matches the average joint motion described in Section 5.2.1. The axis of rotation has been estimated as 65° from the horizontal plane and 33° from the midline. These results are plausible, but a better correlation should have been obtained because only the “best” positions from the path of motion were used to construct the axis of rotation. Both simulations lack magnitude in yaw, and the large pitch has also been identified as problematic. I suspect these problems are caused by the optimization method failing to find the “best” contact position. In the future, a more robust optimization method should be experimented with. I would also like to see experiments that determine whether the contact area estimation is too discrete, and the deformation distance is noise like. An investigation of these measures will likely provide insight into the best optimization method of use. Based on these results, further work is required to validate the biomechanics model of joint motion.
5.2.4 A discussion of simulation parameters

The simulation's parameters are shown in Figure 33, Figure 35, and Figure 37. The step parameters of the *FletcherReeves* module are the most significant towards getting continuous well defined patterns in results. The voxel size has been chosen for the *horizontal*, *depth*, and *quadratic* steps because contact area significantly changes as we step and a new file or row of voxels are included in the intersection. The *roll*, *pitch*, and *yaw* steps are not obvious because the rotation for a significant change in contact area is dependant on the geometry of the object studied. Rotations of $3^\circ$ to $4^\circ$ have been found successful through several experiments.

The problem of discontinuity near the neutral position has been attributed to alignment of volume data sets. The six *initial* parameters of the *FletcherReeves* module can be modified to reduce this behavior. Non-zero values which are less than the step have been chosen and used.
Chapter 6

Conclusion

A new biomechanical model for predicting joint motion directly from one medical scan has been designed and implemented. The simulation has been tested in two environments, and patterns in joint motion have been found similar to my own scientific observations; however, results are not in agreement with biomechanics literature. Problems have been attributed to the optimization method, and further work will be required to verify the biomechanical model of joint motion. The goal of this research has not been attained. Although results lend some support to the model, they are not strong enough to verify it.

The major accomplishments of this research are:

- Development of tools to further biomechanical research.
- Presentation of a method to construct spatially enumerated models from volume data.
- Introduction of new algorithms to estimate volume of intersection and contact area.
- Use of volume and contact area measurements to simulate joint motion.
- Evaluation of a proposed biomechanics model for predicting joint motion, resulting in some specific areas for future work.

Another attainment of this work, is as a case study in biomechanical computing. Biomechanics presents unique problems; however despite discrepancies in results, volume techniques seem to be an appropriate method to solve them.

6.1 Limitations

The modeling process has involved a substantial amount of research and development;
however, I speculate that only the minimum requirements of this problem have been satisfied. First, the resolution of the acquired data is not high enough to differentiate the two cartilage structures from one another (synovial fluid should be visible between the two). Furthermore, an unsuitable image processing algorithm has been applied to the scanned data during acquisition that has further complicated segmentation. The image acquisition should be improved because “the single easiest method for improving a visualization system is to put better data into it” [Ney 1993].

The second problem is achieving an accurate segmentation with a minimum amount of user time. The semi—automatic segmentation algorithms often result in “holes” or “spill,” and determining appropriate parameters can be time consuming because it is often a trial and error process. The nature of magnetic resonance data is also a hindrance to accurate segmentation because bone is not homogeneous. It is comprised of two statistically different distributions termed cortical (the shell or surface) and trabecular (the interior). The mean density value of cortical bone is less than the mean density value of trabecular bone; therefore, only the most frequent type of bone will be segmented. The painting algorithm can be used to “touch up” results to produce an accurate segmentation that is entirely dependent on the user’s knowledge of radiology. Unfortunately, the painting algorithm is time consuming and tedious to use; particularly, because data interpolation produces additional slices that must be segmented.

Currently, the simulation is limited to non—interactive studies, and this is a major limitation for a clinical application. The volume approach to motion simulation requires large computing (one minute for each contact evaluation on a 100 MHz Silicon Graphics workstation) and memory resources (62 MB to load all modules and a medical data set); thus, the time required to execute the simulation is the primary limitation; each simulation of Chapter 5 required a full day to execute on a Silicon Graphics Challenge series server. However, the system was developed by making as few assumptions as possible. When presented with a choice of a restricted optimized algorithm against a generic algorithm, the generic algorithm was chosen. The execution speed of the simulation can also be
greatly improved by making use of parallel or distributed computing resources. This has been partially exploited by developing the simulation as a series of modules that can be run on separate machines or processors; however, the parallelism within the individual modules remains to be exploited.

Furthermore, the research is currently limited to studying joints with primarily one degree of freedom; however, only minor changes are required to study other joints. Joints with two degrees of freedom (for example, the thumb carpometacarpal joint) could be studied by modifying the software to iterate across two dimensions instead of one (the step constraint). In this way, the simulation would produce a grid of optimum positions, where Hamilton's criteria would be applied to each cell. A level set of Hamilton's criteria in the grid then defines the path of motion which might be elliptical or some other two dimensional shape. Similarly, the simulation could be extended to handle other joints with multiple degrees of freedom.

The implementation is also limited to joint configurations such that translation can be used for rigid compression. This assumption may not be appropriate for studying other joints, but in the case of the subtalar joint, it was made to eliminate one dimension from the optimization search. Other joints could be studied by including the vertical component in the optimization process.

Lastly, cartilage is used to simulate joint motion, but bone should also be considered because the contact surface area is affected by cartilage to bone intersection. This problem has not been addressed in this research. In the future, constructive solid geometry techniques could be used to modify the shape of the cartilage by adding or subtracting voxels where appropriate.

6.2 The Biomechanical Study

This thesis has concentrated on aspects of joint motion simulation with little discussion of
its future application in biomechanics. This topic is briefly discussed in this section. Biomechanical joint motion synthesis will be applied to compare range of motion between a patient’s fractured and un-fractured subtalar joint.

These displaced intra-articular fractures occur disproportionately often among young, active workers and constitute about 1% of all fractures. An intra-articular fracture commonly involves the three subtalar joint facets as well as the calcaneo-cuboid joint. Of concern is the relatively poor functional result of the state of the art treatments of calcaneal fractures. Most studies report satisfactory outcomes in less than 75% of the patients. Poor results are all too frequent and are debilitating for many years after the injury and continue to be associated with pain, partial loss of subtalar movement, and footwear problems.

The new technique presented in this work will be used to assess the subtalar joint shortly after reduction, and hopes to improve patient outcome after an injury, since further manipulation of bony fragments may be performed to correct a possible misalignment. Furthermore, a more comprehensive knowledge of the reconstructed calcaneal surface will provide a better knowledge base to improve the outcome of such reductions. It is suggested that the congruity of the subtalar joint surfaces is an important factor influencing recovery. A quantifiable definition of congruity may also be a boon to disparate areas of both basis and applied research including: bone re-modeling, prosthetic design, diarthrodial joint tribology, and arthritic pathologies.

Patients with a diagnosis of a calcaneal fracture, and those that choose a closed reduction will be considered for inclusion in this experiment. The patients who consent to participation will then proceed to have a magnetic resonance image taken of both the damaged and contralateral (opposite) joint before weight bearing commences. It is expected that people who show a good recovery after a displaced intra-articular calcaneal fracture will have had a post-reduction congruity pattern closer to their contralateral limb than those people who do not show a good recovery.
6.3 Future Work

Results are not in agreement with biomechanics literature, but I suspect the problem lies in the optimization method failing to find the "best" contact position. In the future, a more robust optimization method should be experimented with. I would also like to see experiments that determine whether the contact area estimation is too discrete, and the deformation distance is noise like. An investigation of these measures will likely provide insight into the best optimization method of use.

A number of improvements could also be made to improve the running efficiency of the simulation. First, memory and computation time can be reduced through the use of an octree spatial data structure. Currently, data is spatially encoded in a uniform grid, and I speculate that most of the simulation's time is utilized processing either empty or non-participating voxels. Furthermore, as mentioned previously, the parallelism within the individual modules should be exploited. The Silicon Graphics compiler provides options to enable this facility, but it has not been pursued in this work.

The most tedious process for a user is segmenting the medical data because the algorithms provided do not successfully identify objects of interest. In the medical visualization literature, there are other semi-automatic algorithms to perform this task by: combined edge detection with region growing, statistical pattern recognition, or model based hierarchies. Even the implementation of an interactive contouring algorithm to manually outline individual objects would be useful.

Next, the implementation supports only magnetic resonance data from a General Electric Signa series scanner, and inevitably other data formats need to be provided for because the original data has such poor contrast between bone, cartilage, and synovial fluid. Already patients have been recalled and re-scanned using a different device with a new image format. This format will need to be supported in the future.
Finally, but by no means the last enhancement, the biomechanics researcher will need to compare range of motion over time, and visualization techniques to assist this activity have yet to be developed.
Bibliography


ics and Applications, Volume 8, Number 3 (May 1988), 48–68.


